IMS/ESA

Customization Guide

Version 6

SC26-8732-05

IMS/ESA

IBM

Customization Guide

Version 6

SC26-8732-05

Note

Before using this information and the product it supports, be sure to read the general information under “Notices” on page xxi.

Sixth Edition (July 2000) (Softcopy format only)

This edition replaces and makes obsolete the previous edition SC26–8732–04. This edition is available in softcopy format only. The technical changes for this edition are indicated by a vertical bar to the left of a change.

Order hard copies of the second edition of this book and other IBM publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

A form for readers’ comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM Corporation, BWE/H3

P.O. Box 49023

San Jose, CA, 95161-9023

U.S.A.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Notices

Programming Interface Information

This book is intended to help programmers customize an IMS system. This book primarily documents Product-Sensitive Programming Interface and Associated Guidance Information provided by IMS.

Product-Sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning IMS. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-Sensitive programming interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed to run with new product releases or versions, or as a result of service.

However, this book also documents General-Use Programming Interface and Associated Guidance Information provided by IMS.

General-Use programming interfaces allow the customer to write programs that obtain the services of IMS.

General-Use Programming Interface and Associated Guidance Information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

General-Use Programming Interface

General-Use Programming Interface and Associated Guidance Information...

End of General-Use Programming Interface

IMS provides no macros that allow a customer installation to write programs that use the services of IMS.

Attention: Do not use as programming interfaces any IMS macros.

Trademarks

Product Names

In this book, the following licensed programs have shortened names:

- “COBOL for MVS & VM” is referred to as “COBOL”.
- “DB2 for MVS/ESA” is referred to as “DB2”.
- “PL/I for MVS & VM” is referred to as “PL/I”.

Preface

Customization is the task of using supplied services and built-in facilities to enhance or extend a program. This book describes how programmers can use IMS-supplied services and facilities to customize an IMS/ESA (hereafter called IMS) system.

This book contains information about IMS-supplied routines and services that you can use to customize your IMS database, system, and transaction management systems. For the most part, these IMS-supplied routines should meet the needs of your facility. If you have special requirements, however, this book provides you with programming details to help you write routines that customize IMS to function the way you want.

This book is intended for the administrator of the database and transaction management system of IMS and the system programmer of the host system and supported logical units.

Summary of Contents

Part 1 provides general guidelines for customizing IMS routines.

Parts 2, 3, and 4 provide guidance for customizing your IMS system and writing:

- IMS Database Manager (Part 2)
- System (Part 3)
- Transaction Manager (Part 4)

Chapter 1 provides guidelines about writing exit routines; this information applies to Parts 2, 3, and 4. Information on how to use the callable services that IMS provides to perform certain functions is included in this chapter.

Within each of the Parts, the exit routines are organized alphabetically according to the routine's functional name (such as "Command Authorization Exit Routine"). The sequence does not represent the order in which these sections should be read.

For the latest version of the source code for the sample IMS-supplied exit routines, see IMS.DBSOURCE, IMS.SVSOURCE, and IMS.TMSOURCE libraries. The source code is no longer provided in this book as the most current copy is available in the SOURCE libraries that IMS provides.

Part 5 describes the Automated Operator Program Interface (AOI). AOI enables an application program to issue IMS operator commands and to receive responses. It also enables an exit routine to monitor activity and take appropriate action.

Part 6 describes the Data Propagator Interface log record formats that IMS uses to store captured data for asynchronous propagation.

Part 7 describes the External Subsystem Attach facility that enables IMS application programs executing in IMS dependent regions to access data resources that are managed by other (external) MVS subsystems.

Part 8 includes the following appendixes:

Appendix A. DBCTL Function Requests
Appendix B. IMS Support for Synchronous DB2 to IMS Data Propagation

A bibliography and index follow the appendixes.

Prerequisite Knowledge

Before using this book, you need to have a working knowledge of IMS, MVS and its system generation, data communications, and the access methods used by IMS.

Terminology and Related Information

For definitions of terminology used in this book and references to related information in other IMS publications, see *IMS/ESA Master Index and Glossary*.

Change Indicators

Technical changes are indicated in this publication by a vertical bar (|) to the left of the changed text. If a figure has changed, a vertical bar appears to the left of the figure caption.

Summary of Changes

Changes to the Current Edition of This Book

This edition, which is in softcopy only, includes technical and editorial changes to the previous editions.

Information on CQS client exit routines and CQS user exit routines can be found in *IMS/ESA Common Queue Server Guide and Reference*.

Changes to This Book for V6

This book contains new and changed information about the following line items and small programming enhancements:

- Shared Queues/Shared EMH
- Generic Resources
- IMS Sysplex Communication
- Command Authorization Exit Change
- DBCTL Single Cycle Commit
- ETO ENH: DFSINTX0 Change to General DC Exit
- Non-Discardable Messages
- Common Queue Server (Minor changes)

Information in the chapter ″Database Control (DBCTL) Interface″ that appeared in earlier editions of *IMS/ESA Customization Guide* is now in an appendix titled ″Using the Database Resource Adapter″ in *IMS/ESA Application Programming: Database Manager*.

Library Changes for Version 6

The IMS/ESA Version 6 library differs from the IMS/ESA Version 5 library in these major respects:

- *IMS/ESA Common Queue Server Guide and Reference*

  This new book describes the IMS Common Queue Server (CQS).
- *IMS/ESA DBRC Guide and Reference*

  This new book describes all the functions of IMS Database Recovery Control (DBRC).
- The IMS Application Programming summary books (*IMS/ESA Application Programming: Database Manager Summary*, *IMS/ESA Application Programming: Transaction Manager Summary*, and *IMS/ESA Application Programming: EXEC DLI Commands for CICS and IMS Summary*) are no longer included with the IMS library.
- The Softcopy Master Index is not included.
- All information about IRLM 1.5 and data sharing using IRLM 1.5 has been removed from the IMS V6 books. If you use IRLM 1.5, and want to migrate to using IRLM 2.1 and Sysplex data sharing, see *IMS/ESA Release Planning Guide*.
- The chapter that was titled ″Database Control (DBCTL) Interface″ in the *IMS/ESA Customization Guide* has been revised for Open Database Access (ODBA) and moved to ″Appendix A, Using the Database Resource Adapter (DRA)″ in the *IMS/ESA Application Programming: Database Manager*.

Part 1. General Guidelines for Writing IMS Routines

Chapter 1. Guidelines for Writing IMS Exit Routines

This chapter provides guidelines for writing IMS exit routines, a section about the callable services that IMS provides to enable exit routines to perform certain functions, and a section about the callable service return and reason codes.

In this Chapter:

Introduction

This chapter contains general guidelines for writing any exit routine that customizes IMS. Guidelines for writing Database Manager, IMS system, and Transaction Manager (TM) exit routines are provided in this chapter along with sections about the callable services that IMS provides and callable services return and reason codes. At the end of the chapter is a list of IMS-supplied programs for you to refer to when writing your own exit routines. (See Table 4 on page 33.)

Recommendation: Write IMS exit routines in assembler language rather than high-level languages. IMS does not support exit routines running under Language Environment for OS/390. If an exit routine is written in a high- level language executing in the Language Environment for OS/390, the user might incur performance problems or abends. Language Environment for OS/390 is designed for applications running in key 8, problem program state. Most IMS exit routines execute in the IMS control region in key 7 supervisor state. An abend in one of these exit routines causes the IMS control region to abend.

What You Can Customize

Some ways you can customize IMS:

- To edit messages
- To check security
- To edit transaction code input, message switching input, and physical terminal input and output
- To do additional clean-up
- To initialize dependent regions
- To control the number of buffers the RECON data sets use
- To keep track of segments that have been updated

You can write or include additional routines to customize your IMS system.

Sample exit routines are provided in either IMS.DBSOURCE, IMS.SVSOURCE, or IMS.TMSOURCE. Default routines are provided for these exit routines.

Related Reading: For information on how to avoid impacting MVS system integrity, see *MVS/ESA Conversion Notebook*.

Guidelines

You can replace a default exit routine that does not meet your needs by writing one of your own. If you use IMS macros in your exit routine, you must reassemble the routine with the current release level macro library.

Naming the Routines

Using standard MVS conventions, each routine can have any name up to eight characters in length. Be sure that this name is unique and that it does not conflict with the existing members of the data set into which you place the routine. (Because most IMS-supplied routines begin with the prefix “DFS”, “DBF”, “DSP”, or “DXR”, you cannot choose a name that begins with these letters. Also, specify one entry point for the routine.

Related Reading: Refer to *IMS/ESA Administration Guide: Database Manager* for more information on establishing naming conventions.

Naming requirements or exceptions that are specific to an exit routine are noted in the “Naming the Routine” section of each exit routine chapter.

Changeable Interfaces and Control Blocks

The interfaces that IMS supplies for use by the exit routines, including the ISWITCH macro, may change in future releases of IMS. IMS control blocks such as the SCD, PST, DMB, or VTCB may also change. Therefore, if you write an exit routine that uses these services or control blocks, you may need to change or reassemble the routine accordingly when you migrate to a new release of IMS.

IMS Standard User Exit Parameter List

Many of the IMS user exit routines use a standard user exit interface. This interface allows the exit routines to use callable services to access IMS blocks. At the same time, this interface creates a clearly differentiated programming interface (CDPI) between IMS and the exit routine. Part of the interface consists of a standard user exit parameter list. The list contains items such as a pointer to a version number and a pointer to a function-specific parameter list. All standard user exit parameter lists that have the same version number will contain the same parameters. If a new parameter is added, it is added to the end of the parameter list and the version number is incremented.

There are currently two active versions of the standard user exit parameter list. The first version (version 1) contains only pointers to the version number and the function-specific parameter list. The second active version (version 3) contains pointers to the version number, a callable services token, a 256-byte work area, the function-specific parameter list, and the address of a user data table (if applicable). If an exit routine is written to use a parameter that was added in a later version, and the exit routine can execute in an environment in which earlier versions of the parameter list could be received, the exit routine should check the version of the parameter list it received to ensure that the data is available to the exit routine.

Any exit routine that uses the standard user exit parameter list that does not use the latest version of the parameter list will note which version of the parameter list it supports.

Using the ISWITCH Macro

The ISWITCH macro changes execution from the dependent region TCB to the control or DL/I address space. ISWITCH also exits cross-memory mode.

ISWITCH must have addressability to the SCD and, for the ISWITCH example below, to the PST. The address of the SCD is obtained from the PSTSCDAD field in the PST.

For Fast Path exit routines, specify TO=CTL.

```
ISWITCH TO=DLI,ECB=PSTDECB
SLR    R1,R1           Get a zero
ST     R1,PSTDECB      Clean ECB after target memory post
LTR    R15,R15         Successful?
BNZ    ERR1            No
```

When a Fast Path exit routine issues an ISWITCH to the control region, it must issue a second ISWITCH call specifying TO=DEP to return to the dependent region before returning to the caller of the exit routine. This is done only in an exit routine that is entered from a Fast Path module.

```
ISWITCH TO=DEP,ECB=PSTDECB
SLR    R1,R1           Get a zero
ST     R1,PSTDECB      Clean ECB after target memory post
LTR    R15,R15         Successful?
BNZ    ERR1            No
```

Exit routines should not use ISWITCH TO=RET, because unpredictable results may occur. (ISWITCH TO=RET could be used in previous IMS releases.) Ensure that all instances of ISWITCH TO=RET are changed to ISWITCH TO=DEP.

Link-Editing the Routines

Most modules receive control and must return control in AMODE=31, and must be able to execute in cross-memory and TASK modes.

Recommendation:RMODE=ANY is recommended.

Recommendation: All TM exit routines can be entered simultaneously by multiple dispatchable tasks. Therefore, it is highly recommended that all TM exit routines are coded as *reentrant* (RENT).

All routines receive control and must return control in 31-bit addressing mode (AMODE 31) and must be able to execute in RMODE ANY and AMODE 31.

If an exit routine is link-edited as reentrant (RENT), it must be truly reentrant (for example, it cannot depend on **any** information from a previous iteration and it cannot store into itself).

If an exit routine is link-edited as reusable (REUSE), it must be truly reusable (it cannot depend on *any* information in itself from a previous iteration), but it can depend on information that it saves in the specific block passed to it. If you link-edit a routine that is serially reusable, it must be used for a single database only.

If an exit routine is link-edited as neither RENT nor REUSE, it can store into itself and depend on the information saved in the block that is passed to it.

An exit routine that is link-edited as reentrant is loaded in key 0 storage to automatically protect the exit routine from being accidentally or intentionally modified.

Guidelines

Specific requirements and exceptions are noted in each chapter of Parts 1, 2, and 3. Refer to the section on “Link-Editing the Routine” included in each exit routine chapter.

Writing IMS Routines That Access Control Blocks

If you use the online change function, you must modify exit routines that depend on control blocks being in contiguous storage. Control blocks for databases, programs, transactions, and routing codes might not be contiguous after an online modification. This applies specifically to:

DMB Directory Entries (DDIR)

PSB Directory Entries (PDIR)

If your routine accesses IMS control blocks, you can find DSECTs for these blocks in the following macros:

Macro	**DSECT**
ISCD	System content directory (SCD)
DFSDDIR	DMB Directory entry (DDIR)
DFSPDIR	PSB Directory entry (PDIR)
DFSDMB	Data management block (DMB)
DFSPSB	Program specification block (PSB)
DBFESCD	Extended system content directory (ESCD)
DBFRCTE	Routing code table entry (RCTE)

Writing ETO Exit Routines

Unless otherwise stated, all non-LU 6.2 exit routines are available to terminals defined both statically at system definition and dynamically using the Extended Terminal Option (ETO) feature. Some exit routines are loaded at initialization if ETO=Y. (If this is the case, it is noted in each chapter in the section on link-editing or including the routine.) While these exit routines are only loaded if the ETO feature is used, they are available for use by static and dynamic ACF/VTAM terminals.

Related Reading: For more information on ETO, see *IMS/ESA Administration Guide: Transaction Manager*.

Writing APPC/IMS Exit Routines

The LU 6.2 Edit exit routine (DFSLUEE0) is only available to LU 6.2 devices. The Message Control Error exit routine (DFSCMUX0), the Conversational Abnormal Termination exit routine (DFSCONE0), the Transaction Authorization exit routine (DFSCTRN0), and the Fast Path Routine for Input Edit/Routing exit routine (DBFHAGU0) also support LU 6.2 devices. No other exit routines support LU 6.2 devices or are affected by APPC/IMS.

Saving Registers

IMS exit routines need to save registers in the save area pointed to by Register 13. This save area is provided at entry. In general, the save area passed to the exit is in 31-bit storage. You should save and restore registers in 31-bit mode.

There are two types of save areas that exit routines use to save registers:

- A prechained save area passed to the exit routine by IMS or the calling application

- A single save area used by exit routines that use the Version 3 standard user exit parameter list

Using the Prechained Save Area

IMS or the application that calls the exit routine passes a prechained save area to the exit. The routine must step forward to the next save area in the save area set before processing any data.

The save area address given to the exit routine has a prechained forward save area pointer at offset 8 and a prechained backward pointer at offset 4. The exit routine can use the forward save area pointed to by offset 8 but **must not** alter the first three words of the save area.

Before returning control to IMS, the routine must step back to the original save area and restore IMS registers.

Using the Single Save Area

When an exit routine uses the version 3 standard user exit parameter list, it does not receive a prechained save area. Instead, the routine points to a single save area in register 13. The exit routine must use this save area to save registers from IMS or the calling application.

If the exit routine calls other applications or routines, including IMS Callable Services, the routine must provide an additional save area. The 256-byte dynamic work area passed to exit routines that use the version 3 parameter list can be used as one or more save areas.

Before returning control to IMS, the exit routine must restore the registers to IMS or the calling application. For more information on the standard user exit interface refer to “IMS Standard User Exit Parameter List” on page 4.

Considering Cross-Memory

IMS exit routines can be entered in cross-memory mode. Do not issue any SVC (except ABEND) or I/O request.

If the routine runs in the DL/I address space and you need to perform a function that cannot be done in cross-memory mode, issue an ISWITCH TO=DLI to exit cross-memory. Because of the overhead in performing a task switch from the dependent address space to the IMS control program, use ISWITCH infrequently. ISWITCH TO=DLI is not valid for TM exit routines.

If you are not using the DL/I address space option, execution after the ISWITCH continues in the control address space. With LSO=S, execution continues in the DL/I address space. TO=DLI on ISWITCH performs the correct switching in all environments.

With LSO=S, DL/I exits cannot address data in the control address space.

Most terminal-related control blocks are not addressable from the DL/I address space.

Considering Performance

Efficiency of exit routines is a prime concern for IMS performance. Most routines are called from the IMS control region and get control in key 7 supervisor state. Some routines might be called from mainline processing code running under the

Guidelines

IMS Control Region task. The amount and type of processing that is done by those routines can directly contribute to the total path length and time required to complete a unit of work. Other units of work that must wait to run under a task currently in use by an exit routine can also be impacted. An abend in a exit routine that executes in the IMS control region can cause the IMS control region to abend.

Recommendation: Code user-written routines in ways that minimize path length and processing time as much as possible.

Services such as OS WAITs, SVCs, and I/O can all contribute to poor performance and should be used sparingly. When an IMS callable service exists, it is recommended that you use it, rather than the MVS equivalent, because the callable service is usually optimized to perform more efficiently in an IMS sub-dispatching environment.

Recommendation: Code user exit routines only in assembler. Exit routines written in other languages might functionally work, but unacceptable performance can result. For example, using a high- level language runtime environment (such as Language Environment for OS/390) can result in unacceptable performance with the initialization and termination of the environment.

Using IMS Callable Services

IMS Callable Services are services provided by IMS for use by IMS exit routines. These services provide clearly defined interfaces that allow exit routines to request various functions.

Types of Callable Services

There are three types of IMS Callable Services: storage services, control block services, and automated operator interface (AOI) services.

Storage services supports four functions:

- Get storage
- Free storage
- Load module
- Delete module

Control block services supports two functions:

- Find control block
- Scan control block

AOI services supports three functions:

- Insert message
- Enqueue message
- Cancel message

Exit Routines Eligible for Callable Services

Table 1 on page 9 shows the exit routines that are eligible for callable services and the types of callable service that they can use. See the exit routine chapters for more information on how an exit routine uses callable services.

1. DFSAOE00 is the only exit routine eligible to use AOI callable services.

Table 1. Exits and Associated Callable Services

	Callable Services		
Exit name	**Storage**	**Control Block**	**AOI**
DBFHAGU0	X	X	
DFSAOE00 [1]	X		X
DFSAOUE0	X	X	
DFSBSEX0	X		
DFSCCMD0	X	X	
DFSCMLR0	X	X	
DFSCMLR1	X	X	
DFSCMPR0	X	X	
DFSCMPX0	X	X	
DFSCMTR0	X	X	
DFSCNTE0	X	X	
DFSCONE0	X	X	
DFSCSGN0	X	X	
DFSCSMB0	X	X	
DFSCTRN0	X	X	
DFSCTSE0	X	X	
DFSCTTO0	X	X	
DFSFEBJ0	X	X	
DFSFLGX0	X		
DFSGMSG0	X		
DFSGPIX0	X	X	
DFSINSX0	X	X	
DFSINTX0	X	X	
DFSI7770	X	X	
DFSLGFX0	X	X	
DFSLGNX0	X	X	
DFSME000	X	X	
DFSNDMX0	X		
DFSNPRT0	X	X	
DFSME127	X	X	
DFSO7770	X	X	
DFSPIXT0	X	X	
DFSPPUE0	X		
DFSQSPC0	X	X	
DFSSGFX0	X	X	
DFSSGNX0	X	X	
DFSSIML0	X	X	
DFSS7770	X	X	
DFS29800	X	X	
DFS36010	X	X	

How to Use Callable Services

To use a callable service, do the following:

1. Link your exit routine to the callable service interface module (DFSCSI00).
2. Initialize callable services for your exit routine (CALL DFSCSII0) each time your exit routine gets control.
3. Initialize the callable services parameter list.
4. Initialize the function-specific parameter list.
5. Invoke the callable service (CALL DFSCSIF0).

Repeat steps 3 through 5 as many times as necessary while your exit routine has control.

Not all exit routines perform all five of the preceding steps. See the section called “Using IMS Callable Services” in the description of a specific exit routine to see which steps apply.

Before looking at each of these steps in more detail, let’s look briefly at how IMS Callable Services work.

How Callable Services Work

Exit routines must activate IMS Callable Services in AMODE 31.

The callable services interface module DFSCSI00 contains two entry points that your exit routine can activate: DFSCSII0 and DFSCSIF0.

Entry point DFSCSII0 initializes callable services. To activate initialization, issue CALL DFSCSII0 with the appropriate registers initialized. DFSCSII0 returns a callable services token and a parameter list address. The callable services token must be passed to IMS when you activate one of the callable services. The parameter list address directs reentrant programs to a storage area in which to build parameter lists needed to activate callable services.

Entry point DFSCSIF0 activates one of the callable services. To activate a callable service, issue CALL DFSCSIF0 with the appropriate information specified. You must tell IMS which service to activate. You do this by initializing two parameter lists. The first list, the callable services parameter list, contains information needed by callable services to route the request to the appropriate service. The second list, the function-specific parameter list, defines which service is to be used and provides information required by that service.

When your exit routine receives control back from callable services, register 15 contains a return code indicating whether the call was successful. The callable services parameter list contains a return code and a reason code if the call did not complete successfully. The function-specific parameter list may contain data from a specific callable service.

Generating Parameter Lists in Your Exit Routine: You can use the following assembler macros in your exit routine. These macros generate the parameter list DSECTs you need.

Macro	Description
DFSCSIPL	Generates the DFSCSPL, DFSCSTRG, DFSCCBLK, and DFSAOI parameter list DSECTS for an exit routine.

Macro	Description

Macro	Description	
	DFSCSPL	Generates the callable services parameter list DSECT (CSPARMS)
	DFSCSTRG	Generates the storage services function-specific parameter list DSECT (CSSTRG)
	DFSCCBLK	Generates the control block services function-specific parameter list DSECT (CSBLK)
	DFSAOI	Generates the AOI services function-specific parameter list DSECT (DFSAOI)

Step 1: Linking Your Exit Routine to DFSCSI00

To use callable services, your exit routine must be linked with the callable service interface module DFSCSI00. For some exit routines, this module is linked automatically by IMS. For others, you need to manually link this module to your exit routine.

Automatic Linking: The following exit routines are automatically linked to DFSCSI00 by IMS.

DBFHAGU0	DFSI7770
DFSCMLR0	DFSME000
DFSCMPR0	DFSME127
DFSCNTE0	DFSQSPC0
DFSCONE0	DFSSIML0
DFSCSGN0	DFSS7770
DFSCTRN0	DFS29800
DFSFEBJ0	

Manual Linking: To use callable services, you must manually link these exit routines to DFSCSI00.

DFSAOE00	DFSINSX0
DFSAOUE0	DFSINTX0
DFSBSEX0	DFSLGFX0
DFSCCMD0	DFSLGNX0
DFSCMTR0	DFSNDMX0
DFSCSMB0	DFSNPRT0
DFSCTSE0	DFSPPUE0
DFSGMSG0	DFSSGFX0
DFSGPIX0	DFSSGNX0

Typically, you must manually link DFSCSI00 if your exit routine is a stand-alone module (not linked as part of another IMS load module). When you perform this link-edit, include an ENTRY link-edit control statement that specifies the entry point of your exit routine. This ensures that your exit routine, and not DFSCSI00, receives control from IMS when it is called.

Step 2. Initializing IMS Callable Services (DFSCSII0)

Most exit routines must initialize callable services. To do this, issue a call to entry point DFSCSII0. DFSCSII0 returns a callable services token and a parameter list address.

Callable Services

The callable services token is used to request a specific callable service through a subsequent call to entry point DFSCSIF0.

The parameter list that is returned in register 1, contains the callable services token. You need to extract the token and save it, so it does not get overlayed. Then the parameter list can be formatted for your callable service request. The parameter list is large enough to contain the parameter lists that accompany your request.

Communicating with IMS: IMS uses the entry registers, parameter list, and exit registers to communicate with your exit routine. The contents of register 0 are not preserved on entry and exit.

Content of Registers on Entry to DFSCSII0:

Register	Content
1	ECB Address. On entry, IMS gives the address of an ECB to each exit routine that can issue callable service requests. The ECB address must be passed on the DFSCSII0 initialization call. See the section for each exit routine to determine where to find the ECB address for that exit routine.
13	Address of save area for use by DFSCSII0.
14	Caller’s return address.
15	DFSCSII0 entry point address.

Content of Registers on Return from DFSCSII0:

Register **Content**

1 Address of parameter list.

Offset	Description
0	Callable services token, which is four bytes long.

15 Return Code

Return Code	Meaning
0	Request was successful.
4	Callable services are unavailable.
8	Callable services are unavailable. Initialization failed due to insufficient storage.
12	Callable services are unavailable. Initialization failed due to errors in IMS control blocks.

Step 3. Initializing the Callable Services Parameter List

CSPARMS is the callable services parameter list required for all callable service requests. Callable services use parameters in the list to route control from the module requesting the service to the service routine that processes the request. The list is also used to pass return and reason codes from the service to the exit routine.

Initialize the parameter list with the callable services token and the code of the callable service you want to use (storage services, control block services, or AOI services). All other fields should be cleared. If your exit routine issues multiple calls, you can save the callable services token in a register and restore it to CSPLTOKN on subsequent calls.

Initialize the following fields:

Field	Offset	Length	Description
CSPLTOKN	0	4	IMS Callable Services token
CSPLSERV	4	4	IMS Callable Service code. The values are: 1 = storage services 2 = control block services 3 = AOI services

Step 4. Initializing the Function-Specific Parameter List

After specifying which service you want to use in the callable services parameter list, indicate which function of the service you want to use by initializing the appropriate function-specific parameter list.

The function-specific parameter lists are described in:

- "Requesting IMS Callable Storage Services" on page 14
- "Requesting IMS Callable Control Block Services" on page 17
- "Requesting IMS Callable AOI Services" on page 21

Step 5. Activating an IMS Callable Service (DFSCSIF0)

To activate a callable service, issue:

```
CALL DFSCSIF0
(callable services parameter list, function-specific
parameter list)
```

Communicating with IMS: IMS uses the entry registers, parameter lists, and exit registers to communicate with your exit routine.

Content of Registers on Entry to DFSCSIF0:

Register	Content
1	Address of two-word parameter list built by CALL macro.

Offset	Description
0	Callable services parameter list address
4	Function-specific parameter list address

Register	Content
13	Address of save area for use by DFSCSIF0
15	DFSCSIF0 entry point address

Content of Registers on Return from DFSCSIF0:

Register	Content
15	Return Code

Return Code	Meaning
0	Request successful
4	Request unsuccessful

If the request is unsuccessful, refer to the return (CSPLRTRN) and reason code (CSPLRESN) fields in the callable services parameter list.

Field	Description
CSPLRTRN	Return code set with error codes defined in DFSCSPL. For a list of these codes, refer to “Return Codes (CSPLRTRN)” on page 23.
CSPLRESN	Reason code set with error codes defined in DFSCSPL. For a complete description of the reason codes, refer to one of the following sections: If CSPLRTRN= 4, refer to “Callable Service Interface Reason Codes (CSPLRESN)” on page 23. If CSPLRTRN = 8 or 20, refer to “Function-Specific Parameter List Reason Codes (CSPLRESN)” on page 23.

Requesting IMS Callable Storage Services

CSSTRG is the function-specific parameter list used for IMS Callable Storage Service requests. It is defined by the DFSCSTRG macro.

The function-specific parameter list contains the information that storage services need to perform the function you requested (get or free storage, load or delete a module). The function-specific parameter list is also used to return data to the exit routine.

You must initialize the function-specific parameter list for storage services before calling DFSCSIF0 to activate storage services. All fields that are not used as input to DFSCSIF0 should be cleared.

Using the GET Storage Function

GET storage is the storage service for obtaining user storage. The storage can be obtained in private storage or CSA with either doubleword or page boundary alignment. The storage can be requested above (31-bit) or below (24-bit) the 16-megabyte line.

To request the GET storage function, initialize the following fields in the function-specific parameter list (CSSTRG):

Field	Offset	Length	Description
CSSTFUNC	0	4	IMS storage service function code value: 1 = GET storage
CSGTLEN	4	4	Length of storage to obtain
CSGTSP	8	4	Storage subpool identifier values: 0 = private storage X'FFF' = CSA storage
CSGTLOC	C	4	Storage location identifier values: 0 = 31-bit storage 1 = 24-bit storage
CSGTBNDY	10	4	Storage boundary identifier values: 0 = doubleword boundary 1 = page boundary

The following field (in CSSTRG) is returned from the GET storage function:

Field	Offset	Length	Description
CSGTADDR	14	4	Storage address

Using the FREE Storage Function

FREE storage is the storage service used to release user storage previously obtained by the GET storage service. The requestor specifies the address of the storage. The storage subpool (private or CSA) specified on the FREE request must be the same value specified on the GET request.

To request the FREE storage function, initialize the following fields in the function-specific parameter list (CSSTRG):

Field	Offset	Length	Description
CSSTFUNC	0	4	IMS storage service function code value: 2 = FREE storage
CSFRSTAD	4	4	Address of storage to release
CSFRLEN	8	4	Length of storage to release
CSFRSP	C	4	Storage subpool identifier values: 0 = private storage X'FFF' = CSA storage

No data is returned from the FREE storage service.

Using the LOAD Module Function

LOAD module is the storage service used to load a module. The module can be loaded in private storage or CSA. The module can be loaded above (31-bit) or below (24-bit) the 16-megabyte line. The name of the module must be specified. If the module was loaded previously but you want a new copy of the module, you can request a load of a new copy.

The LOAD module function can be requested by callers running in cross memory mode. In this case, the LOAD module function determines if the primary address space is either CTL or DLI/SAS, and ensures that the call executes in the proper address space in non-cross memory mode. The LOAD module function restores the cross memory environment before returning control to the caller.

There might be a noticeable performance impact for cross memory callers issuing the LOAD module function, because this call requires that the environment be switched from cross memory mode to non-cross memory mode and then restored. Use of the LOAD module function should be kept to a minimum for mainline path exit routines.

To use the LOAD module function, initialize the following fields in the function-specific parameter list (CSSTRG):

Field	Offset	Length	Description
CSSTFUNC	0	4	IMS storage service function code value : 5 = LOAD module
CSLDNAME	4	8	Name of module to load

Field	Offset	Length	Description
CSLDSP	C	4	Storage subpool identifier values: 0 = private storage X'FFF' = CSA storage
CSLDLOC	10	4	Module storage location identifier values: 0 = 31-bit storage 1 = 24-bit storage
CSLDUSE	14	4	Module reuse identifier values: 0 = use existing copy of module if found 1 = load a new copy of module

The following fields are returned from the LOAD module function:

Field	Offset	Length	Description
CSLDMEP	18	4	Module entry point
CSLDMLEN	1C	4	Module length bit 0 is set to one when the module was previously loaded

Using the DELETE Module Function

DELETE module is the storage service used to delete a module previously obtained by the LOAD storage service. The requester specifies either the module name or module address. If more than one copy of the module was loaded, the address should be used instead of the name to ensure that the correct copy is deleted. The module storage subpool (private or CSA) specified on the DELETE request must be the same value specified on the LOAD request.

The DELETE module function can be requested by callers running in cross memory mode. In this case, the DELETE module function determines if the primary address space is either CTL or DLI/SAS, and ensures that the call executes in the proper address space in non-cross memory mode. The DELETE module function restores the cross memory environment before returning control to the caller.

There might be a noticeable performance impact for cross memory callers issuing the DELETE module function, because this call requires that the environment be switched from cross memory mode to non-cross memory mode and then restored. Use of the DELETE module function should be kept to a minimum for mainline path exit routines.

To request the DELETE module function, initialize the following fields in the function-specific parameter list (CSSTRG):

Field	Offset	Length	Description
CSSTFUNC	0	4	IMS storage service function code value: 6 = DELETE module
CSDLNAME	4	8	Name of module to delete. Either module name or module address must be specified to delete a module. The unused field should be cleared. If the module name is not specified, this field should be cleared and CSDLEP must be specified.

Field	Offset	Length	Description
CSDLEP	C	4	Address of module to delete. If more than one copy of the module was loaded, delete the module by specifying the module entry point. This ensures that the correct copy of the module is deleted. If both name and address are specified, the module is deleted using the address. If the address is not given, the name must be specified and all copies will be deleted.
CSDLSP	10	4	Storage subpool identifier values: • 0 = private storage • X'FFF' = CSA storage

No data is returned from the DELETE module function.

Requesting IMS Callable Control Block Services

CSCBLK is the function-specific parameter list used for IMS Callable Control Block Service requests. It is defined by the DFSCCBLK macro.

The function-specific parameter list contains the information control block services needed to perform the function you requested (find or scan a control block). The function-specific parameter list is also used to return data to the exit routine.

You must initialize the function-specific parameter list for control block services before calling DFSCSIF0 to activate control block services. All fields that are not used as input to DFSCSIF0 should be cleared.

Using the FIND Control Block Function

FIND control block is used to find a specific instance of a control block. The search type identifies the type of control block to locate. A search type may include more than one type of control block. A list of the search types is in the description of the CSFDTYPE field in the following table. The control block name or identifier is used to find a specific instance of the control block.

Initialize the function-specific parameter list before calling DFSCSIF0 to activate control block services. All fields that are not used as input to DFSCSIF0 should be cleared.

Initializing the Function-Specific Parameter List for FIND: In all instances, you need to initialize the following two fields:

Field	Offset	Length	Description
CSCBFUNC	0	4	IMS control block services function code value: 1 = FIND control block

Field	Offset	Length	Description
CSFDTYPE	4	4	Control block search type values: 1 = FIND CCB 2 = FIND CNT, or LNB 3 = FIND RCNT 4 = FIND CNT, LNB, or RCNT 5 = FIND SPQB 6 = FIND VTCB 7 = FIND CNT descriptor 8 = FIND LOGON descriptor 9 = FIND USER descriptor 10 = FIND transaction

Depending on the type of block you want to find, you must initialize the following fields:

To Find	Initialize

CCB	Specify one of the following fields. Clear the unused field.
	`CSFDEID = EBCDIC CCB identifier`
	`CSFDBID = Binary CCB identifier`

CNT or LNB	Use the LTERM name to locate a specific CNT or LNB. `CSFDNAME = LTERM name`
RCNT	Use the LTERM name to locate a specific RCNT. `CSFDNAME = LTERM name`
CNT, LNB, or RCNT	Use the LTERM name to locate a specific CNT, LNB, or RCNT. `CSFDNAME = LTERM name`
SPQB	Use the USER name to locate a specific SPQB. `CSFDNAME = USER name`

VTCB	Either the node name alone or the node and user name can be used to locate a specific VTCB. If the user name is not used on the request, the field should be cleared.
	`CSFDNODE = Node name`
	`CSFDUSER = User name`

CNT, LOGON, or USER Descriptor	Specify the name of the descriptor you want to locate. `CSFDNAME = CNT, LOGON, or USER descriptor name`
Transaction	Specify the transaction code you want to find. `CSFDNAME = Transaction name`

Output Returned from FIND Control Block Services: Depending on the type of search specified, one of the following is returned in the CSFDBLKA field in the function- specific parameter list:

Search Type	Output from Service
FIND CCB	CCB address
FIND CNT or LNB	CNT or LNB address
FIND RCNT	RCNT address
FIND CNT, LNB, or RCNT	CNT, LNB, or RCNT address
FIND SPQB	SPQB address
FIND VTCB	CLB address
FIND CNT descriptor	USRD address
FIND LOGON descriptor	CLB address

Search Type	**Output from Service**
FIND USER descriptor	USRD address
FIND Transaction	SMB address FIND transaction also returns the PDIR address in field CSFCBLK2.

Using the SCAN Control Block Function

SCAN control block is used to scan control blocks of a certain type. The first time the SCAN function is activated, the current control block address should be 0. SCAN returns the first control block that meets the search criteria. The SCAN function may be activated subsequently to locate additional control blocks. Subsequent searches start where the previous scan left off.

On subsequent SCAN requests, the current block address is passed back to the service. The search starts with the current control block to locate the next control block meeting the criteria. The blocks are not retrieved in alphabetic sequence.

Qualifying the Scan: To further qualify the scan, a generic name or a name containing wild cards can be specified for CNT, LNB, RCNT, SPQB, and VTCB control block types.

- A generic name consists of one or more characters of the name followed by an asterisk. Generic names must be padded with blanks.

  For example, assume valid names are DFSAAAAA, DFSZZZZZ, and DFSABBBB. Multiple scan requests using the generic name 'DFSA*' may be used to obtain the control block addresses for DFSAAAAA and DFSABBBB. In this case, DFSZZZZZ would not be returned to the requester.
- A wild card character is represented by the '%' character. One or more wild cards can replace characters within the name when that position in the name can be any character.

  For example, assume valid names are DFSAABBB, DFSZZBBB, and DFSABCDE. Multiple scan requests using the name DFS%%BBB containing wild card characters in positions 4 and 5 would return control block addresses for DFSAABBB and DFSZZBBB. DFSABCDE would not be returned to the requester.

You must initialize the function-specific parameter list before calling DFSCSIF0 to activate control block services. All fields that are not used as input to DFSCSIF0 should be cleared.

Initializing the Function-Specific Parameter List for SCAN: To request a SCAN and search type, you always need to initialize the first two fields as follows:

Field	Offset	Length	Description
CSCBFUNC	0	4	IMS control block service function code value: 2 = SCAN control block

Field	Offset	Length	Description
CSSCTYPE	4	4	Control block search type indicator values: 1 = SCAN CCB 2 = SCAN CNT or LNB 3 = SCAN RCNT 4 = SCAN CNT, LNB, or RCNT 5 = SCAN SPQB 6 = SCAN VTCB 7 = not used 8 = SCAN LOGON descriptor 9 = SCAN USER descriptor 10 = not used

Depending on the type of search you want, you may also need to initialize one or more of the following fields in the function-specific parameter list.

To Scan / **Initialize**

CCB
: Specify whether you want to scan for the first CCB or to start the scan at the current CCB.

```
CSSCCBLK = Current CCB address or zero
```

CNT or LNB
: Specify whether you want to scan for the first CNT or LNB or to start the scan at the current CNT or LNB. Use the LTERM name to narrow the scope of the scan. If the LTERM name is not used, clear the field.

```
CSSCCBLK = Current CNT or LNB address or zero
CSSCNAME = LTERM name
```

RCNT
: Specify whether you want to scan for the first RCNT or to start the scan at the current RCNT. Use the LTERM name to narrow the output of the scan. If the LTERM name is not used, clear the field.

```
CSSCCBLK = Current RCNT address or zero
CSSCNAME = LTERM name
```

CNT, LNB, or RCNT
: Specify whether you want to scan for the first CNT, LNB, or RCNT, or to start the scan at the current CNT, LNB, or RCNT. Use the LTERM name to narrow the output of the scan. If the LTERM name is not used, clear the field.

```
CSSCCBLK = Current CNT, LNB, or RCNT address,
           or zero
CSSCNAME = LTERM name
```

SPQB
: Specify whether you want to scan for the first SPQB or to start the scan at the current SPQB. Specify the USER name to narrow the output of the scan. If the USER name is not specified, clear the field.

```
CSSCCBLK = Current SPQB address or zero
CSSCNAME = USER name
```

VTCB
: Specify whether you want to scan for the first VTCB or to start the scan at the current CLB. Specify either the NODE name alone, or the NODE and USER name to narrow the output of the scan. If the name fields are not specified, clear the fields.

```
CSSCCBLK = Current CLB address or zero
CSSCNODE = NODE name
CSSCUSER = USER name
```

LOGON Descriptor
: Specify whether you want to scan for the first LOGON descriptor or to start the scan at the current LOGON descriptor.

```
CSSCCBLK = Current LOGON descriptor address
           or zero
```

USER Descriptor	Specify whether you want to scan for the first USER descriptor or to start the scan at the current USRD. `CSSCCBLK = Current USRD address or zero`

Output Returned from SCAN Control Block Services: Depending on the type of scan specified, one of the following is returned in the CSSCNBLK field in the function-specific parameter list:

Search type	Output from service
SCAN CCB	Next CCB address
SCAN CNT or LNB	Next CNT or LNB address
SCAN RCNT	Next RCNT address
SCAN CNT, LNB, or RCNT	Next CNT, LNB, or RCNT address
SCAN SPQB	Next SPQB address
SCAN VTCB	Next CLB address
SCAN LOGON descriptor	CLB address of next LGND
SCAN USER descriptor	Next USRD address

Requesting IMS Callable AOI Services

DFSCAOI is the function-specific parameter list used for IMS Callable AOI Service requests. It is defined by the DFSCAOI macro.

The function-specific parameter list contains the information that AOI services needs to perform the function you requested (insert, enqueue, or cancel a message). The function-specific parameter list is also used to return data to your exit routine.

You must initialize this function-specific parameter list before calling DFSCSIF0 to activate AOI callable services. All fields that are not used as input to DFSCSIF0 should be cleared.

Using the INSERT Function

The INSERT function inserts the first, or a subsequent, message segment into a message buffer. The message segments are not available to the AO application until an enqueue is issued specifying an AOI token.

To request the INSERT function, initialize the following fields in the function-specific parameter list:

Field	Offset	Length	Description
CAOIFUNC	0	4	IMS AOI service function code value: 1 = INSERT message segment
CAOIDMTK	4	4	Directed message token
CAINMSEG	8	4	Address of message segment

Using the ENQUEUE Function

The ENQUEUE function inserts the last or only message segment into the message buffer. It enqueues this message segment to the AOI token the requester has specified. Enqueue then makes the entire message available to the AO application. If ENQUEUE is requested with a message segment address of 0, all previously inserted message segments are made available to the AO application.

To request the ENQUEUE function, initialize the following fields in the function-specific parameter list (DFSCAOI):

Field	Offset	Length	Description
CAOIFUNC	0	4	IMS AOI service function code value: 2 = ENQUEUE segment to AOI token
CAOIDMTK	4	4	Directed message token
CAENMSEG	8	4	Address of message buffer
CAENTCNT	12	4	Number of AOI token names in the token list addressed by the next word in this parameter list
CAENTLST	16	4	Address of a token list. Each 12-byte entry in the list contains the following: **+0 =** The 8-byte alphanumeric AOI token name to which the message is to be enqueued **+8 =** The 4-byte code from the ENQUEUE function indicating whether the message was successfully enqueued to the AOI token. Possible codes are: **0 =** Enqueue to AOI token was successful. **1 =** Enqueue was unsuccessful. AOI token name was blanks. **2 =** Enqueue was unsuccessful. AOI token name contained invalid characters. **3 =** Enqueue was unsuccessful. Enqueue could not get AOIP storage. **4 =** Enqueue was unsuccessful. An internal latch error occurred.

Using the CANCEL Function

The CANCEL function cancels messages that have been inserted into the message buffer but not yet enqueued to the AOI token. Canceled messages are not made available to the application program.

To request the CANCEL function, initialize the following fields:

Field	Offset	Length	Description
CAOIFUNC	0	4	IMS AOI service function code value: 3 = CANCEL message segments
CAOIDMTK	4	4	Directed message token

Callable Service Return and Reason Codes

This section describes the return and reason codes for IMS Callable Services. All return and reason codes are in hexadecimal format.

Return Codes (CSPLRTRN)

Return codes are in field CSPLRTPN in the callable services parameter list. Following are the return codes indicating why the request did not complete successfully:

Return Code	Meaning
4	Callable service interface error. A callable service interface error occurred. The service request was not processed.
8	Function-specific parameter list error. While processing the callable service request, an error occurred in the function-specific parameter list.
20	Service processing error. An error occurred while processing the callable service request. The error may be a user error or an internal system error.

Callable Service Interface Reason Codes (CSPLRESN)

Following are the reason codes when return code field CSPLRTRN = 4:

Reason Code	Meaning
4	Callable services token is 0. The field CSPLTOKN in the callable services parameter list DFSCSPL is 0.
8	Callable services token is invalid. The field CSPLTOKN in the callable services parameter list DFSCSPL does not contain a valid callable services token.
C	Service code is not specified. The field CSPLSERV in the callable services parameter list DFSCSPL is 0.
10	Service code is invalid. The field CSPLSERV in the callable services parameter list DFSCSPL does not contain a valid callable service code. The service code is too large.
14	Service is not supported. The field CSPLSERV in the callable services parameter list DFSCSPL contains a value for a callable service code that is not supported in the current environment or is a reserved function.
30	Function code is not specified. The function code field in the function-specific parameter list is 0.
34	Function code is invalid. The function code field in the function-specific parameter list contains a function code that is too large.
38	Function is not supported. The function code field in the function-specific parameter list contains a value for a callable service function code that is not supported in the current environment or is a reserved function.

Function-Specific Parameter List Reason Codes (CSPLRESN)

Following are the reason codes when return code field CSPLRTRN = 8 or 20: The reason codes are described by service and by function.

GET Storage Service Reason Codes

Following are the reason codes for GET function parameter errors:

When CSPLRTRN = 8:

Reason Code	Meaning
4	Invalid subpool parameter. The field CSGTSP in the function-specific parameter list DFSCSTRG contains an invalid subpool value.
8	Invalid location parameter. The field CSGTLOC in the function-specific parameter list DFSCSTRG contains an invalid storage location value.

Reason Code	Meaning
C	Invalid boundary parameter. The next CSGTBNDY in the function-specific parameter list DFSCSTRG contains an invalid storage boundary value.
10	Length parameter not specified. The field CSGTLEN in the function-specific parameter list DFSCSTRG is 0.

When CSPLRTRN = 20: If you receive any reason code not listed below, please contact IBM Service.

Reason Code	Meaning
06 00 00 04	Storage could not be allocated.
06 00 00 08	Parameter list error.

FREE Storage Service Reason Codes

Following are the reason codes for FREE function parameter errors:

When CSPLRTRN = 8:

Reason Code	Meaning
4	Invalid subpool parameter. The field CSFRSP in the function-specific parameter list DFSCSTRG contains an invalid subpool value.
8	Address parameter not specified. The field CSFRSTAD in the function-specific parameter list DFSCSTRG is 0.
C	Length parameter not specified. The field CSFRLEN in the function-specific parameter list DFSCSTRG is 0.

When CSPLRTRN = 20: If you receive any reason code not listed below, please contact IBM service.

Reason Code	Meaning
07 00 00 04	Storage was not released. A value in the second byte of the reason code is provided by the associated MVS Service. For example, the 04 in the second byte of reason code 07 04 00 04 is returned from MVS FREEMAIN. Additional information may be found in the *IMS/ESA Messages and Codes*, Appendix C, IMODULE FREESTOR Return Codes.
07 00 00 08	Parameter list error.
07 00 00 0C	Unable to locate storage descriptor block. Storage address may be invalid or storage subpool specification may be incorrect.

LOAD Storage Service Reason Codes

Following are the reason codes for LOAD function parameter errors:

When CSPLRTRN = 8:

Reason Code	Meaning
4	Invalid subpool parameter. The field CSLDSP in the function-specific parameter list DFSCSTRG contains an invalid subpool value.
8	Invalid location parameter. The field CSLDLOC in the function-specific parameter list DFSCSTRG contains an invalid module location value.
C	Invalid use parameter. The field CSLDUSE in the function-specific parameter list DFSCSTRG contains an invalid module reuse value.

Reason Code	Meaning
10	Name parameter not specified. The field CSLDNAME in the function-specific parameter list DFSCSTRG does not contain a module name.
14	The caller is running in cross memory mode, and the primary address space is not CTL or DLI.

When CSPLRTRN = 20: If you receive any reason code not listed below, please contact IBM Service.

Reason Code	Meaning
02 00 00 04	Module was not found.
02 00 00 08	DFSMODU0 allocation error.
02 00 00 0C	BLDL/FETCH allocation error.
02 00 00 10	FETCH/BLDL I/O error occurred loading the requested module.
02 00 00 24	DCB was not open for BLDL.
02 00 00 28	Caller was authorized, but module was found in unauthorized library.

DELETE Storage Service Reason Codes

Following are the reason codes for DELETE function parameter errors:

When CSPLRTRN = 8:

Reason Code	Meaning
4	Invalid subpool parameter. The field CSDLSP in the function-specific parameter list DFSCSTRG contains an invalid subpool value.
8	Name and address was not specified. The field CSDLNAME in the function-specific parameter list DFSCSTRG does not contain a module name, and CSDLEP does not contain a module address.
C	The caller is running in cross memory mode, and the primary address space is not CTL or DLI.

When CSPLRTRN = 20: If you receive any reason code not listed below, please contact IBM Service.

Reason Code	Meaning
04 00 00 04	Module was not found.
04 00 00 0C	Module storage was not released.

FIND Control Block Service Reason Codes

Following are the reason codes for FIND function parameter errors:

When CSPLRTRN = 8:

Reason Code	Meaning
4	FIND type was not specified. The field CSFDTYPE in the function-specific parameter list DFSCCBLK is 0.
8	FIND type was invalid. The field CSFDTYPE in the function-specific parameter list DFSCCBLK does not contain a valid control block search type value. The search type value is too large.
C	CCBID was not specified. The field CSFDEIB in the function-specific parameter list DFSCSTRG does not contain an EBCDIC CCB identifier, and CSFDBID does not contain a binary CCB identifier.

Return and Reason Codes

Reason Code	Meaning
10	Control block name was not specified. The field CSFDNAME in the function-specific parameter list DFSCCBLK does not contain a name.

When CSPLRTRN = 20: Following are the reason codes you may get when searching CCB, CNT, LNB, RCNT, SPQB, and USER descriptor control block types:

Reason Code	Meaning
4	Block was not found.
40 00 00 00	CBTS latch held, cannot process request.

Following are the reason codes you may get when searching VTCB and LOGON descriptor control block types:

Reason Code	Meaning
4	Cannot find CLB with VTAM CID or node/descriptor name.
8	NO VTCBs/LGNDs are in system.
40 00 00 00	CBTS latch held, cannot process request.

The following are the reason codes that may be encountered when searching for a transaction control block type.

Reason Code	Meaning
8	Transaction was not found.
40 00 00 00	CBTS latch held, cannot process request.

SCAN Control Block Service Reason Codes

Following are the reason codes for SCAN function parameter errors:

When CSPLRTRN = 8:

Reason Code	Meaning
4	SCAN type was not specified. The field CSSTYPE in the function-specific parameter list DFSCCBLK is 0.
8	SCAN type was invalid. The field CSSCTYPE in the function-specific parameter list DFSCCBLK does not contain a valid control block search type value. The search type value is too large or is a reserved function.

When CSPLRTRN = 20: Following are the reason codes you may get when searching CCB, CNT, LNB, RCNT, SPQB, and USER descriptor control block types:

Reason Code	Meaning
4	End of queue was found.
8	No block is in system.
14	Bad INUSE call. Verify that the CSSCCBLK and CSSCNAME fields are properly initialized.
18	Bad NOUSE call. Verify that the CSSCCBLK and CSSCNAME fields are properly initialized.

Reason Code	Meaning
40 00 00 00	CBTS latch held, cannot process request.

Following are the reason codes you may get when searching VTCB and LOGON descriptor control block types:

Reason Code	Meaning
4	Cannot find VTCB matching arguments.
8	No VTAM nodes were in system.
40 00 00 00	CBTS latch held, cannot process request.

INSERT AOI Service Reason Codes

Following are the reason codes for INSERT function parameter errors:

When CSPLRTRN = 8:

Reason Code	Meaning
4	Directed message token was 0.
8	Directed message token was invalid.
C	Message segment address was 0.
10	Message segment length (LL field) was 0.

When CSPLRTRN = 20:

Reason Code	Meaning
4	IMS could not get the storage required to process the call.

ENQUEUE AOI Service Reason Codes

Following are the reason codes for ENQUEUE function parameter errors:

When CSPLRTRN = 8:

Reason Code	Meaning
4	Directed message token was 0.
8	Directed message token was invalid.
10	Message segment address was specified, but segment length (LL field) was 0.
14	AOI token count field was 0.
18	AOI list token address was 0.
1C	One or more tokens was processed successfully.
20	No tokens were processed successfully.

CANCEL AOI Service Reason Codes

Following are the reason codes for CANCEL function parameter errors:

Return and Reason Codes

When CSPLRTRN = 8:

Reason Code	Meaning
4	Directed message token was 0.
8	Directed message token was invalid.
C	No message exists to cancel.

Example of a Callable Services Request

The following is an example of how an exit routine could use IMS Callable Services. In the example, the storage returned from DFSCSII0 is divided into three areas. These areas are for the parameter lists used for the call to DFSCSIF0. The first area is used for the MVS CALL parameter list, the second for the IMS Callable Service parameter list, and the third for the function specific parameter list. The labels, CSICLLEN and CSPLPLEN, used in the examples are defined as EQU statements in the macro DFSCSIPL. These labels represent the length of the MVS parameter list built by the CALL macro and the length of the IMS Callable Services parameter list.

```
***********************************************************************
*                                                                     *
*       --------------------------------                              *
*       GETSTOR - GET STORAGE SUBROUTINE                              *
*       --------------------------------                              *
*                                                                     *
*       THIS SUBROUTINE INVOKES IMS CALLABLE SERVICES TO              *
*       GET WORKING STORAGE.  THE CALLER PASSES THE REQUIRED          *
*       STORAGE LENGTH.  THE SUBROUTINE THEN OBTAINS PRIVATE,         *
*       31-BIT STORAGE ON A DOUBLEWORD BOUNDARY.                      *
*                                                                     *
*                                                                     *
*       INPUT REGISTERS:                                              *
*          R8  = REQUESTED STORAGE LENGTH                             *
*          R9  = ECB ADDRESS                                          *
*          R10 = LINKAGE REGISTER                                     *
*                CALLED BY BAL 10,GETSTOR                             *
*                                                                     *
*       OUTPUT REGISTERS:                                             *
*          R1  = STORAGE ADDRESS                                      *
*          R9  = ECB ADDRESS                                          *
*          R10 = LINKAGE REGISTER                                     *
*          R15 = RETURN CODE                                          *
*                0        - CALL COMPLETED SUCCESSFULLY               *
*                NON-ZERO  - STORAGE REQUEST FAILED                   *
*                            RETURN CODE FROM IMS CALLABLE STORAGE    *
*                            SERVICES - GET STORAGE FUNCTION          *
*                                                                     *
*       REGISTER USAGE:                                               *
*          R0  = WORK REGISTER                                        *
*          R1  = WORK REGISTER                                        *
*          R2  = IMS CALLABLE SERVICE TOKEN                           *
*          R3  = IMS CALLABLE SERVICES PARAMETER LIST                  *
*          R4  = IMS STORAGE SERVICES PARAMETER LIST                  *
*          R5  = MVS CALL PARAMETER LIST                              *
*          R8  = REQUESTED STORAGE LENGTH                             *
*          R9  = ECB ADDRESS                                          *
*          R14 = WORK REGISTER                                        *
*          R15 = WORK REGISTER                                        *
*                                                                     *
***********************************************************************
GETSTOR  DS    0H
         SPACE
```

```
************************************************************************
*       INVOKE CALLABLE SERVICES INITIALIZATION ENTRY POINT           *
*       DFSCSII0, TO OBTAIN THE CALLABLE SERVICE TOKEN AND             *
*       PARAMETER LIST STORAGE.                                        *
************************************************************************
         LR    1,9                ECB ADDRESS
         CALL  DFSCSII0           INVOKE INIT ENTRY POINT
         LTR   15,15              CALL SUCCESSFUL?
         BNZ   GSTREXIT           NO, ERROR RETURN
         SPACE
************************************************************************
*       R1 CONTAINS A PARAMETER LIST ADDRESS.                          *
*       OFFSET 0 IN THE LIST CONTAINS THE 4-BYTE CALLABLE              *
*       SERVICE TOKEN.  EXTRACT THE TOKEN FROM THE PARAMETER           *
*       LIST FOR USE ON THE GET STORAGE REQUEST.                       *
************************************************************************
         LR    5,1                COPY STORAGE ADDRESS
         L     2,0(,5)            CALLABLE SERVICE TOKEN
         SPACE
************************************************************************
*       R5 CONTAINS THE ADDRESS TO USE FOR THE PARAMETER               *
*       LIST FOR THE MVS CALL MACRO.  USING THE EQU LABELS             *
*       IN MACRO DFSCSIPL, CARVE THE STORAGE RETURNED BY               *
*       DFSCSII0 INTO SEPARATE PARAMETER LISTS TO BE USED              *
*       ON THE CALL TO DFSCSIF0.                                       *
************************************************************************
         LA    3,CSICLLEN(,5)     CALLABLE SERVICE PARM LIST ADDR
         LA    4,CSPLPLEN(,3)     STORAGE SERVICES PARM LIST ADDR
         SPACE
************************************************************************
*       PARAMETER LIST RETURNED FROM DFSCSII0 HAS BEEN CARVED INTO     *
*       THREE PARTS:                                                   *
*                                                                      *
*       R5                R3                     R4                    *
*        -------------------------------------------------------      *
*       |  MVS CALL AREA  |  IMS CALL SVC AREA  |  STG SVC AREA  |      *
*        -------------------------------------------------------      *
*                                                                      *
************************************************************************
         SPACE
************************************************************************
*       INITIALIZE CALLABLE SERVICE PARAMETER LIST.                    *
*                                                                      *
*       ENTIRE LIST IS CLEARED SO ALL RESERVED AND NON-INPUT           *
*       FIELDS (SUCH AS THE RETURN AND REASON CODES)                   *
*       ARE SET TO ZERO.  THE CALLABLE SERVICE CODE IS                 *
*       INITIALIZED TO REQUEST STORAGE SERVICES                        *
*       AND THE CALLABLE SERVICE TOKEN IS SAVED IN THE LIST.           *
************************************************************************
         USING CSPARMS,3           CALLABLE SERVICES PARM LIST DSECT
         XC    CSPARMS(CSPLPLEN),CSPARMS  CLEAR CALLABLE SERVICES LIST
         LA    0,CSPLSTRG          STORAGE SERVICE CODE
         ST    0,CSPLSERV          INSERT SERVICE CODE IN LIST
         ST    2,CSPLTOKN          INSERT CALLABLE SERVICE TOKEN
         SPACE
************************************************************************
*       INITIALIZE STORAGE SERVICE PARAMETER LIST                      *
*                                                                      *
*       ENTIRE LIST IS CLEARED SO ALL RESERVED AND NON-INPUT           *
*       FIELDS (SUCH AS THE RETURN AND REASON CODES)                   *
*       ARE SET TO ZERO.  THE STORAGE SERVICES                         *
*       FUNCTION CODE IS INITIALIZED TO REQUEST THE GET STORAGE        *
*       FUNCTION.  PARAMETERS ARE INITIALIZED TO OBTAIN 31-BIT,        *
*       PRIVATE STORAGE IN SUBPOOL 0 ON A DOUBLEWORD BOUNDARY.         *
*                                                                      *
************************************************************************
         USING CSSTRG,4            STORAGE SERVICES PARM LIST DSECT
```

Callable Services Example

```
         XC    CSSTRG(CSGTPLEN),CSSTRG  CLEAR STORAGE SERVICES LIST
         LA    0,CSSTGET           GET STORAGE FUNCTION CODE
         ST    0,CSSTFUNC          INIT FUNCTION CODE PARAMETER
         SPACE
         ST    8,CSGTLEN           INIT STORAGE LENGTH PARAMETER
         SPACE
         LA    0,CSGTPRI           PRIVATE STORAGE INDICATOR
         ST    0,CSGTSP            INIT STORAGE SUBPOOL INDICATOR
         SPACE
         LA    0,CSGT31B           31-BIT STORAGE INDICATOR
         ST    0,CSGTLOC           INIT STORAGE LOCATION PARAMETER
         SPACE
         LA    0,CSGTDBLW          DOUBLE WORD BOUNDARY INDICATOR
         ST    0,CSGTBNDY          INIT STORAGE BOUNDARY PARAMETER
         SPACE
***********************************************************************
*       THE CALLABLE SERVICES PARAMETER LIST HAS BEEN INITIALIZED     *
*       TO INVOKE IMS STORAGE SERVICES.  THE STORAGE SERVICES         *
*       PARAMETER LIST HAS BEEN INITIALIZED TO OBTAIN USER STORAGE.   *
*       ISSUE THE IMS CALLABLE SERVICE REQUEST TO OBTAIN STORAGE.     *
***********************************************************************
         CALL  DFSCSIF0,((3),(4)),MF=(E,(5))
         LTR   15,15              STORAGE REQUEST SUCCESSFUL?
         BNZ   GSTREXIT           NO, RETURN TO CALLER
         SPACE
         L     1,CSGTADDR         STORAGE ADDRESS
         SPACE
***********************************************************************
*       RETURN TO CALLER                                              *
***********************************************************************
GSTREXIT DS    0H
         BR    10                 RETURN TO CALLER
         LTORG
         DFSCSIPL
```

Control Block Usage

This section is a directory of the control blocks and their associated fields that are intended for access by exit routines. This section also identifies and specifies limitations on the use of these control blocks. If only certain fields within a control block are intended for customer use, they are listed next to the control block name in the table below. Any field that does not appear next to the control block name is not intended for customer use.

Unless otherwise specified, the only information that is part of the interface for exit routines is the control block name and any specific fields associated with that control block. For a field that is part of the interface, the only information that is part of the interface for exit routines is the named field.

The following control blocks, and their associated fields and flags, are intended for use as, or as part of, a product-sensitive interface. Flags are enclosed in parenthesis next to their associated fields.

Table 2. Control Blocks and Associated Fields and Flags.

Control Block Name	Fields and Flags Intended for Use
CCB	CCBNUMB
CIB	CIBMNAME, CIBDTYP (CIBDNDS)
CLB	CLBNAME, CLBCURR, CLBCNTQB
CNT, LNB	CNTDEQCT, CNTENQCT, CNTNAME, CNTDQCT, CNTCTBPT, CNTCNTPT

Table 2. Control Blocks and Associated Fields and Flags. (continued)

Control Block Name	Fields and Flags Intended for Use
CTB	CTBCTT, CTBTERM, CTBFLAG1 (CTB1SIGN, CTB1PRES), CTBFLAG2 (CTB2LOCK, CTB2TEST, CTB2EXCL), CTBFLAG3 (CTB3SEG1), CTBACTL (CTBAEOM, CTBAINC), CTBFEAT, CTBINCT, CTBOUTCT, CTBCNT, CTBCIBPT, CTBPRSTN, CTBCNTPT, CTBFLAG6 (CTB6SDON, CTB6TRNI), CTBUSID, CTBOUSID
CTT	CTTDEVIC (CTTD3286, CTTDTYP1, CTTDLU4), CTTSEND, CTTEDIT, CTTIEDIT, CTTOPT2 (CTT2DIT), CTTOPT5 (CTT5DYN)
CVB	CVBCCMD
FEIB	FEIBOFLG (FEIBRPQ1, FEIBERP, FEIBTMED), FEIBMSGN, FEIBLTRM, FEIBMSG, FEIBUNID, FEIBNDST, FEIBERPN, FEIBLDST, FEIBULNG FEIBUSER, FEIBIMID
MFSFLDE	FLDFLAG (FLDOPT, FLDEXIT, FLDATTR, FLDEATR), FLDELTH, FLDVECT, FLDLTH, FLDADDR (OPT3LTH, OPT3ID, OPT3DATA)
MFSSEGE	SEGFLAG, SEGOPT (SEGEXIT, SEGECHO), SEGVECT, SEGLTH, SEGFLDRC (SEGDL)
MSNB	MSNFLG1 (MSN1DEQ), MSNFLG3 (MSN3DQND, MSN3DQLM)
PDIR	PDIRSYM, PDIRCODE (PDIRLOCK, PDIRNOSC, PDIRSCHD, PDIRDBST, PDIRBALG), PDIROPTC (PDIRRETN, PDIRGPSB, PDIRDOPT, PDIRPARL, PDIRBAD), PDIRFLG3 (PDIRIFPR, PDIRIFPM, PDIRIFPU)
RCNT	CNTDEQCT, CNTENQCT, CNTNAME, CNTDQCT
SCD	SSCDIMID, SCDQTU, SCDQTL, SCDSSTYP (SCDSSDBC, SCDSSDCC), SSCDIMSR, SSCDIMSL
SMB	SMBDEQCT, SMBENQCT, SMBTRNCD, SMBSTATS (SMBSRESP, SMBSMULT, SMBSNOQU, SMBNOSC, SMBLOCK, SMBSINQU, SMBSQERR), SMBFLAG1 (SMB1CONV, SMB1UPP, SMBCPIC, SMB1NORE, SMB1INIT), SMBFLAG2 (SMB2DRRT, SMBFPPTX, SMBFPXCL, SMB2SMS, SMB2RMT), SMBFLAG3 (SMB3WFI), SMBPRIOR, SMBCLASS, SMBSPAL, SMBLMTCT, SMBCOUNT, SMBSIDR, SMBSIDL, SMBMXRGN, SMBPARLM, SMBAOIFL (SMBTCMDA, SMBNOSCH), SMBPDIRN, SMBRCTEN
SPQB, USRD	SPQBHSQN

The following table provides a list, by exit, of the control blocks that are intended for use as, or as part of, a product-sensitive interface:

Table 3. Exits and Associated Control Blocks

Exit name	Associated Control Blocks
DBFHAGU0	SCD
DBFHC40	none
DBFHDC44	none
DBFLHSH0	none
DBFHUMSE1	none
DFSAOEE0	none
DFSAOUE0	CLB, CTB, SCD
DFSBSEX0	none

Table 3. Exits and Associated Control Blocks (continued)

Exit name	Associated Control Blocks
DFSCCMD0	CLB, CTB, CTT, CVB, SCD
DFSCMLR0	PDIR, SCD, SMB
DFSCMLR1	none
DFSCMPR0	SCD
DFSCMPX0	none
DFSCMTR0	CLB, CTB, CTT, PDIR, SCD, SMB
DFSCMUX0	MSNB
DFSCNTE0	CLB, CNT, CTB
DFSCONE0	CCB, CTB, PDIR, SCD, SPQB, SMB
DFSCSGN0	CTB, SCD
DFSCSMB0	CLB, CTB
DFSCTRN0	CLB, CNT, CTB, PDIR, SCD, SMB
DFSCTSE0	CNT, CTB, PDIR, SCD, SMB
DFSCTTO0	CLB, CNT, CTB, SCD
DFSFEBJ0	FEIB, PDIR, SMB
DFSFLGX0	none
DFSFTFX0	none
DFSGMSG0	none
DFSGPIX0	PDIR, SMB
DFSINSX0	CLB, SCD
DFSINTX0	CLB, SCD
DFSISIS0	none
DFSI7770	CLB, CNT, CTB, SCD
DFSLGFX0	CLB, SCD
DFSLGNX0	CLB, SCD
DFSLUEE0	none
DFSME000	MFSFLDE
DFSNDMX0	none
DFSNPRT0	CLB, CTB, CTT, PDIR, SMB
DFSME127	MFSSEGE, CLB
DFSO7770	CLB, CTB, CTT, SCD
DFSPIXT0	CTB, PDIR, SMB
DFSPPUE0	none
DFSPRE60	none
DFSPRE70	none
DFSQSPC0	PDIR, SCD, SMB
DFSSBUX0	none
DFSSGFX0	CLB, SCD
DFSSGNX0	CIB, CLB, CTB, CTT, SCD
DFSSIML0	CLB, CNT, CTB, CTT, SCD

Table 3. Exits and Associated Control Blocks (continued)

Exit name	Associated Control Blocks
DFSSS050	none
DFSSS060	none
DFSS7770	CLB, CNT, CTB, CTT, SCD
DFSTXIT0	none
DFSYORU0	none
DFSYIOE0	none
DFSYPRX0	none
DFS29800	CLB, CNT, CTB, PDIR, SCD, SMB
DFS36010	CLB, CTB, SCD
DSPCEXT0	none

Customization Guide Exit Routines

The location of the sample exit routines and programs, and the chapter that describes them, are listed in Table 4.

Table 4. Customization Guide Exit Routines

Exit Routine	Chapter Title	Location
DBFHAGU0	"Chapter 32. Fast Path Input Edit/Routing Exit Routine (DBFHAGU0)" on page 263	IMS.TMSOURCE
DBFHC40/ DBFHDC44	"Chapter 4. Data Entry Database Randomizing Routine (DBFHDC40/ DBFHDC44)" on page 67	IMS.DBSOURCE
DBFLHSH0	"Chapter 5. Data Entry Database Resource Name Hash Routine (DBFLHSH0)" on page 73	IMS.DBSOURCE
DBFUMSE1	"Chapter 6. Data Entry Database Sequential Dependent Scan Utility Exit Routine (DBFUMSE1)" on page 77	
DFSAOE00	"Chapter 11. Type 2 Automated Operator Exit Routine (DFSAOE00)" on page 119	IMS.SVSOURCE
UETRANS	"Chapter 69. Automated Operator (AO) Application Program (GU, GN, CMD, and GCMD Calls)" on page 435	IMS.TMSOURCE
DFSAOPGM	"Chapter 70. Automated Operator (AO) Application Program (GMSG, ICMD, and RCMD Calls)" on page 445	IMS.SVSOURCE
DFSAOUE0	"Chapter 29. Type 1 Automated Operator Exit Routine (DFSAOUE0)" on page 227	IMS.TMSOURCE
DFSBSEX0	"Chapter 30. Build Security Environment Exit Routine (DFSBSEX0)" on page 253	
DFSCCMD0	"Chapter 13. Command Authorization Exit Routine (DFSCCMD0)" on page 135	IMS.SVSOURCE
DFSCKWD0	"Chapter 16. IMS Command Language Modification Facility (DFSCKWD0)" on page 149	IMS.SVSOURCE
DFSCMLR0/ DFSCMLR1	"Chapter 46. MSC Link Receive Routing Exit Routines (DFSCMLR0/DFSCMLR1)" on page 341	IMS.TMSOURCE

Table 4. Customization Guide Exit Routines (continued)

Exit Routine	Chapter Title	Location
DFSCMPR0	“Chapter 47. MSC Program Routing Exit Routine (DFSCMPR0)” on page 345	IMS.TMSOURCE
DFSCMPX0	“Chapter 9. Segment Edit/Compression Exit Routine (DFSCMPX0)” on page 93	IMS.DBSOURCE
DFSCMTR0	“Chapter 60. Terminal Routing Exit Routine (DFSCMTR0)” on page 403	IMS.TMSOURCE
DFSCMTU0	“Chapter 28. User Message Table (DFSCMTU0)” on page 213	
DFSCMUX0	“Chapter 44. Message Control/Error Exit Routine (DFSCMUX0)” on page 327	IMS.TMSOURCE
DFSCNTE0	“Chapter 45. Message Switching (Input) Edit Routine (DFSCNTE0)” on page 337	IMS.TMSOURCE
DFSCONE0	“Chapter 31. Conversational Abnormal Termination Exit Routine (DFSCONE0)” on page 257	IMS.TMSOURCE
DFSCSGN0	“Chapter 59. Sign On/Off Security Exit Routine (DFSCSGN0)” on page 399	IMS.TMSOURCE
DFSCSMB0	“Chapter 63. Transaction Code (Input) Edit Routine (DFSCSMB0)” on page 415	IMS.TMSOURCE
DFSCTRN0	“Chapter 62. Transaction Authorization Exit Routine (DFSCTRN0)” on page 411	IMS.TMSOURCE
DFSCTSE0	“Chapter 55. Security Reverification Exit Routine (DFSCTSE0)” on page 383	IMS.TMSOURCE
DFSCTTO0	“Chapter 53. Physical Terminal (Output) Edit Routine (DFSCTTO0)” on page 373	IMS.TMSOURCE
DFSFDOT0	“Chapter 15. Dump Override Table (DFSFDOT0)” on page 145	
DFSFEBJ0	“Chapter 33. Front-End Switch Exit Routine (DFSFEBJ0)” on page 269	IMS.TMSOURCE
DFSFLGX0	“Chapter 21. Logger Exit Routine (DFSFLGX0)” on page 177	
DFSFTFX0	“Chapter 20. Log Filter Exit Routine (DFSFTFX0)” on page 173	IMS.SVSOURCE
DFSGMSG0	“Chapter 35. Greeting Messages Exit Routine (DFSGMSG0)” on page 287	IMS.TMSOURCE
DFSGPIX0	“Chapter 34. Global Physical Terminal (Input) Edit Routine (DFSGPIX0)” on page 283	IMS.TMSOURCE
DFSHDC40	“Chapter 7. HDAM Randomizing Routines (DFSHDC40)” on page 81	IMS.DBSOURCE
DFSINSX0	“Chapter 51. Output Creation Exit Routine (DFSINSX0)” on page 361	IMS.TMSOURCE
DFSINTX0	“Chapter 36. Initialization Exit Routine (DFSINTX0)” on page 289	IMS.TMSOURCE
DFSISIS0	“Chapter 25. Resource Access Security Exit Routine (DFSISIS0)” on page 205	IMS.SVSOURCE
DFSI7770	“Chapter 66. 7770-3 Input Edit Routine (DFSI7770)” on page 423	IMS.TMSOURCE

Table 4. Customization Guide Exit Routines (continued)

Exit Routine	Chapter Title	Location
DFSLGFX0	"Chapter 40. Logoff Exit Routine (DFSLGFX0)" on page 311	IMS.TMSOURCE
DFSLGNX0	"Chapter 41. Logon Exit Routine (DFSLGNX0)" on page 315	IMS.TMSOURCE
DFSLUEE0	"Chapter 43. LU 6.2 Edit Exit Routine (DFSLUEE0)" on page 321	IMS.TMSOURCE
DFSLULU0	"Chapter 42. LU 6.2 Destination Exit Routine (DFSLULU0)" on page 319	
DFSME000	"Chapter 37. Input Message Field Edit Routine (DFSME000)" on page 293	IMS.TMSOURCE
DFSME127	"Chapter 39. Input Message Segment Edit Routine (DFSME127)" on page 305	IMS.TMSOURCE
DFSNDMX0	"Chapter 22. Non-Discardable Messages Exit Routine (DFSNDMX0)" on page 185	IMS.TMSOURCE
DFSNPRT0	"Chapter 38. Input Message Routing Exit Routine (DFSNPRT0)" on page 297	IMS.TMSOURCE
DFSO7770	"Chapter 67. 7770-3 Output Edit Routine (DFSO7770)" on page 427	IMS.TMSOURCE
DFSPIXT0	"Chapter 52. Physical Terminal (Input) Edit Routine (DFSPIXT0)" on page 369	IMS.TMSOURCE
DFSPPUE0	"Chapter 23. Partner Product Exit Routine (DFSPPUE0)" on page 195	
DFSPRE60	"Chapter 26. System Definition Preprocessor Exit Routine (Input Phase) (DFSPRE60)" on page 207	IMS.SVSOURCE
DFSPRE70	"Chapter 27. System Definition Preprocessor Exit Routine (Name Check Complete) (DFSPRE70)" on page 211	IMS.SVSOURCE
DFSQSPC0	"Chapter 54. Queue Space Notification Exit Routine (DFSQSPC0)" on page 377	IMS.TMSOURCE
DFSSBUX0	"Chapter 10. Sequential Buffering Initialization Exit Routine (DFSSBUX0)" on page 111	IMS.DBSOURCE
DFSSGFX0	"Chapter 57. Signoff Exit Routine (DFSSGFX0)" on page 387	IMS.TMSOURCE
DFSSGNX0	"Chapter 58. Sign-On Exit Routine (DFSSGNX0)" on page 391	IMS.TMSOURCE
DFSSIML0	"Chapter 56. Shared Printer Exit Routine (DFSSIML0)" on page 385	IMS.TMSOURCE
DFSSS050	"Chapter 17. Large SYSGEN Sort/Split Input Exit Routine (DFSSS050)" on page 153	IMS.SVSOURCE
DFSSS060	"Chapter 18. Large SYSGEN Sort/Split Output Exit Routine (DFSSS060)" on page 157	IMS.SVSOURCE
DFSS7770	"Chapter 68. 7770-3 Sign-on Exit Routine (DFSS7770)" on page 431	IMS.TMSOURCE
DFSTXIT0	"Chapter 61. Time-Controlled Operations (TCO) Exit Routine (DFSTXIT0)" on page 407	IMS.TMSOURCE
DFSYDRU0	"Chapter 48. OTMA Destination Resolution Exit Routine (DFSYDRU0)" on page 349	IMS.TMSOURCE

Table 4. Customization Guide Exit Routines (continued)

Exit Routine	Chapter Title	Location
DFSYIOE0	“Chapter 49. OTMA Input/Output Edit Exit Routine (DFSYIOE0)” on page 353	IMS.TMSOURCE
DFSYPRX0	“Chapter 50. OTMA Prerouting Exit Routine (DFSYPRX0)” on page 357	IMS.TMSOURCE
DFS29800	“Chapter 64. 2972/2980 Input Edit Routine (DFS29800)” on page 419	IMS.TMSOURCE
DFS36010	“Chapter 65. 4701 Transaction Input Edit Routine (DFS36010)” on page 421	IMS.TMSOURCE
DSPBUFFS	“Chapter 12. Buffer Size Specification Facility (DSPBUFFS)” on page 131	
DSPCEXT0	“Chapter 24. RECON I/O Exit Routine (DSPCEXT0)” on page 199	IMS.SVSOURCE

Part 2. Database Exit Routines

Chapter 2. Data Capture Exit Routine

This section documents General-Use Programming Interface and Associated Guidance Information. See “Notices” on page xxi to understand this classification of information.

When an application program updates an IMS database with a DL/I insert, replace, or delete call, the updated data is passed and made available to a Data Capture exit routine. You can write a Data Capture exit routine that receives control whenever a segment, for which the exit routine is defined, is updated. Your exit routine processes the data after the DL/I call completes but before control is returned to the application program. The DL/I call is considered complete and the PCB status set when the exit routine is called. Figure 1 shows how control passes among the application, the full-function or DEDB database, and the exit routine.

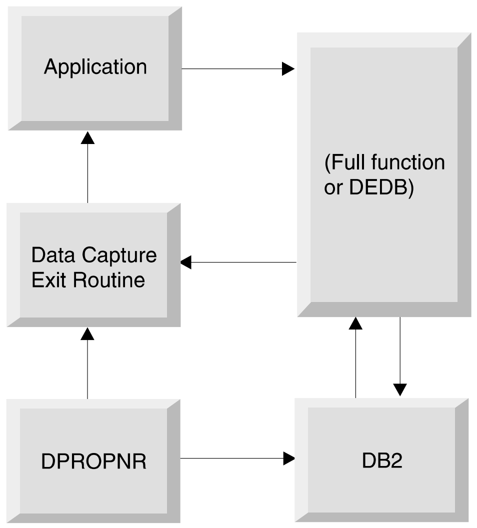


Figure 1. Calling Order with Data Capture

You might want to capture changed data to enable replication of that data to a relational DB2 database as shown in Figure 1.

As an alternative to capturing data synchronously, you can also propagate captured data asynchronously by using either of the following methods:

- Use the logging option on the EXIT= parameter of DBDGEN.
- Use DPROPNR and specify that the data is to be propagated asynchronously.

Related Reading: For more information on asynchronous data capture, see “Chapter 71. Propagating Captured Data Asynchronously” on page 457. For more information about DPROPNR, see *DataPropagator NonRelational MVS/ESA An Introduction*, or visit the DPROPNR home page on the Web at:

http://www.software.ibm.com/data/dpropnr

The following table describes data capture support for IMS environments for both full-function and DEDB databases.

Data Capture Exit Routine

Table 5. Data Capture Support for IMS Environments

	CICS DB/CTL	CICS Batch	IMS Batch	IMS IFP	IMS BMP	IMS MPP
Data Capture Exit EXIT=exitname	No	Yes[1]	Yes	Yes	Yes	Yes
Asynchronous Data Capture EXIT= *, LOG	Yes	Yes[1]	Yes	Yes	Yes	Yes

Note: [1]BATCH is a pure IMS batch environment that is available with CICS DB/CTL (no CICS code executing).

In this Chapter:

About This Routine

The main purpose of capturing updated data and making it available to an exit routine is to propagate the IMS data to the relational environment of DB2. You can write your own exit routine, use a separate product, DataPropagator Nonrelational (hereafter referred to as DPropNR), or write a DPropNR-supported exit routine. If you write your own exit routine, you can code it to perform tasks other than data propagation. The sample Data Capture exit routine provided at the end of this chapter only propagates data.

Restriction: This exit routine cannot be used with CICS, because it conflicts with CICS architecture. (Asynchronous Data Capture does work with DBCTL.) Even though the exit routine works with captured IMS data, CICS cannot use it.

Attributes of the Routine

Regardless of its function, the exit routine must be written in assembler language, C language, COBOL, or PL/I. Routines written in high- level languages running under Language Environment for OS/390 are not supported. A sample Data Capture exit routine in COBOL and a sample in PL/I are provided beginning with"COBOL" on page 53 and "PL/I" on page 56, respectively.

Restriction: Although IMS supports PASCAL and OS/VS COBOL, Data Capture exit routines written in these languages are not supported.

Recommendation:Do not run data capture exit routines under Language Environment for OS/390. Although the data capture exit routine might work running under Language Environment for OS/390, the user might experience performance problems or abends.

Link-Editing the Routine

If you link-edit the exit routine as either RENT or REUSE, it remains in storage until the region terminates as if the exit routine was preloaded. However, non-REUSE exit routines must be loaded each time, because they are deleted from storage after each call.

Loading the Routine

IMS loads the exit routine the first time IMS calls it; preloading the exit routine is not necessary. However, run-time library routines used by high-level languages should be preloaded. After abnormal termination in an IMS Fast Path region (IFP) or in a message processing region (MPP), the exit routine is deleted and must be reloaded. The exit routine must be reloaded when:

- A pseudo or standard abend of the application that is running in the region occurs (regardless of whether the region itself abends along with the application).
- The data capture routine gets an XPCB return code of 16.

Specifying Data Options

In addition to the necessary control information, you can have the following data passed to your exit routine. The data is chained together using pointers.

Physical concatenated key	The fully concatenated key of each segment in the physical hierarchy, including the updated segment. For logical relationships and secondary indexes, this key differs from the key in the PCB feedback area.
Physical segment data	The physical segment updated by the application program, without any PSB field sensitivity.
Data before a replace	The data as it looked before it was updated. Your exit routine must determine what fields the application program changed.
Path data	The physical path data from the root segment to the parent of the updated segment.
Cascade delete data	The data deleted by IMS when an application program deleted a segment that is higher in the hierarchy.

The data is in the same format that was returned to the application program, excluding PSB field sensitivity. For logical children, the segment data follows the logical parent concatenated key. For segments with compression/edit exit routines defined for them, the data is in its expanded or encoded form. For variable-length segments, the first two bytes contain the length ('LL') for the segment.

Related Reading: For more information on these data options, see *IMS/ESA Administration Guide: Database Manager*.

Additional Guidelines

The Data Capture exit routine is called whenever a segment is updated that has the exit routine defined, regardless of the execution environment. The exit routine may use the INQY ENVIRON call to identify the execution environment (batch or online) and determine what functions are available.

Related Reading: For details on using the INQY ENVIRON call, see *IMS/ESA Application Programming: Database Manager*.

The exit routine may issue any DL/I calls allowed by the PSB using the AIB Interface (AIBTDLI). However, any updates that the exit routine makes are not captured and do not call an exit routine.

Data Capture Exit Routine

The Data Capture exit routine is treated as an extension of the application program; IMS attributes SQL or DL/I calls made by the exit routine to the application program. The exit routine and the application run under the same unit of work. SQL and DL/I updates made by the exit routine are committed or aborted along with the application program at sync-point time with the same integrity as the application. The exit routine must follow the same rules as the application program whether the routine makes IMS or DB2 requests.

For data propagation, all DL/I updates must be passed to your exit routine to determine whether to propagate the change to DB2 or not. Both the IMS data and DB2 data must be available and on the same MVS system for either update to occur.

The Data Capture exit routine is called based upon specification in the DBD rather than in the PSB. The exit routine is always called. It is also a global exit routine: once implemented for any segment, all activity in that segment causes IMS to call the exit routine, regardless of which PSB is active. Any performance impact that the exit routine causes occurs across the entire system.

Defining the Routine for Segments

The Data Capture exit routine is specified for a particular segment during DBDGEN. Failure to locate the exit routine during processing results in an application program abend.

DBDGEN supports a new parameter, EXIT=, on the DBD and SEGM statements. If specified on the DBD statement, the parameter applies to all segments within the physical database structure. If specified on the SEGM statement, you may override the specification on the DBD, or can limit the parameter so that only selected segments are propagated when updated. As a SEGM parameter, EXIT= does not apply to other segments; physical children do not inherit the parameters of any of their parents.

You may specify multiple exit routine names, each with different data options, on a single DBD or SEGM statement.

Related Reading: For details on using the EXIT= parameter, see *IMS/ESA Utilities Reference: System*.

Multiple Exit Routines

A single DL/I call might call your exit routine more than once or it might call more than one exit routine. Multiple exit routines are called when there are:

- Multiple exit routines per segment
- Path calls
- Cascade deletes

Multiple exit routines are called in succession before returning to the application program. The sequence depends on the reason multiple exit routines are called:

- Multiple exit routines are defined.

  When multiple exit routines are defined for a single physical segment, the routines are called based on DBDGEN definition order. The first exit routine listed in the DBD or SEGM statement is called, followed by each subsequent exit routine defined for that segment.
- Multiple segments are updated.

When multiple physical segments are updated in a single call, the routines are called in hierarchical order. IMS calls the exit routines for the segments in the same order that the segments were physically updated:

- Top-down for path inserts and path replaces

  Parents must be inserted before dependents. The exit routine for the parent segment must be called before the dependent segment's exit routine.
- Bottom-up for cascade deletes

  The dependent segment's exit routine is called before the parent's exit routine. The root segment's exit routine is called last. If the dependent segment has several exit routines defined for it, they are **all** called at this time. Calling the exit routines in bottom-up order allows propagation to DB2 without requiring referential integrity.

  For each segment type, multiple segment occurrences might be deleted as part of the cascade delete. Each exit routine is called once for each segment occurrence that is deleted. The order the exit is called is the same order in which DL/I deleted the segments.

Using IMS Callable Services with This Routine

This exit is not eligible to use IMS Callable Services.

Communicating with IMS

Each segment that is passed in a dependent region and has the Data Capture exit routine defined for it has two control blocks available for its use. Both the Extended Program Communication Block (XPCB) and the Extended Segment Data Block (XSDB) reside in private storage and have key 8. They are passed to the exit routine according to the AMODE of the exit: above the 16MB line for AMODE 31, and below the 16MB line for AMODE 24.

The order in which the control blocks receive control depends on the type of data updated and passed to the Data Capture exit routine. Figure 2 shows how control flows between the XPCB and the XSDB.

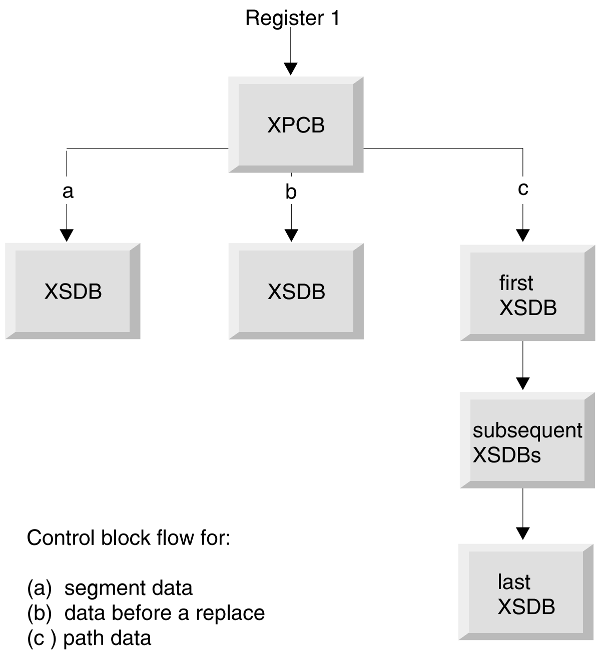


Figure 2. Control Block Flow with Data Capture

Extended Program Communication Block (XPCB)

The XPCB identifies the segment and call functions, provides the address of a work area, and contains additional information that is passed to the exit routine. Every XPCB identifies the physical function performed by DL/I (insert, replace, or delete) and points to the updated data that is passed to the exit routine. Table 6 and Table 7 on page 47 describe the contents of the XPCB. For examples of how to code the XPCB, see “Sample Extended Program Communication Block (XPCB)” on page 51.

For reentrant exit routines, the address of a 256-byte work area is passed in the XPCB. The exit routine can use the work area to save information. One work area exists for each exit routine, and it is initialized to binary zeros the first time the exit routine is given control.

Table 6. XPCB By Offset

OffsetDecimal	Field Name	OffsetDecimal	Field Name	OffsetDecimal	Field Name
0	Eyecatcher	4	Version	6	Release
8	Exit_Name	16	Exit_Return_Code	18	Exit_Reason_Code
20	Database_Name	28	DED_Version_Ptr	32	Segment_Name
40	Call_Function	44	Physical_Function	48	reserved
52	DB_PCB_Ptr	56	DB_PCB_Name	64	INQY_Output_Ptr
68	IO_PCB_Ptr	72	reserved	74	Conc_Key_Length
76	Conc_Key_Ptr	80	Data_SCDB_Ptr	84	Before_XSDB_Ptr
88	Path_XSDB_Ptr	92	reserved	104	Exit_Work_Ptr
108	Null_Ptr	112	reserved	116	CPU Store Clock

Table 7. XPCB Alphabetically

Field Name	Offset Decimal	Data Type	Length Decimal	Field Description
Before_XSDB_Ptr	84	Pointer	4	Address of XSDB for data before it was replaced. Zero if not a physical replace or if data not captured.
Call_Function	40	Character	4	Call used by application to update segment: ISRT, DLET, REPL, or CASC (cascade).
Conc_Key_Length	74	Fixed	2	Length of segment's concatenated key for physical path. Zero if data not captured. Key is optional.
Conc_Key_Ptr	76	Pointer	4	Address of segment's concatenated key for physical path. Zero if data not captured. Key is optional.
CPU_Store_Clock	116	Character	8	CPU time stamp of completion of DL/I call. Obtained from Store Clock instruction.
Database_Name	20	Character	8	Physical database name of updated segment.
Data_XSDB_Ptr	80	Pointer	4	Address of XSDB for segment data. Zero if data not captured.
DBD_Version_Ptr	28	Pointer	4	Address of variable length character string to identify the DBD used for update. First two bytes contain length of string, followed by string itself. String is from DBD VERSION= parameter if was used for DBDGEN. Otherwise, string is date/time of DBDGEN.
DB_PCB_Ptr	52	Pointer	4	Address of database PCB used for DL/I call.
DB_PCB_Name	56	Character	8	The 8-byte name of database PCB used for DL/I call. Null if name not assigned during PSBGEN with the label or PCBNAME= parameter.
Exit_Return_Code	16	Fixed	2	Return code from exit routine.
Exit_Reason_Code	18	Fixed	2	Reason code from exit routine.
Exit_Work_Ptr	104	Pointer	4	Address of 256-byte work area.
Eyecatcher	0	Character	4	'XPCB'
INQY_Output_Ptr	64	Pointer	4	Address of output of an INQY ENVIRON call.
IO_PCB_Ptr	68	Pointer	4	Address of I/O PCB.
Null_Ptr	108	Pointer	4	Zero address for use as null address for languages that do not recognize a zero address as null (such as PL/I).
Path_XSDB_Ptr	88	Pointer	4	Address of XSDB for physical root when path data option requested. XSDBs for path data are chained together, in descending hierarchical order, from physical root to parent of updated segment. Last XSDB has a zero pointer.
Physical_Function	44	Character	4	Physical call function performed: ISRT, DLET, or REPL.
Release	6	Character	2	XPCB release indicator. Along with version, identifies the level of the control block. Current release is R1.
Segment_Name	32	Character	8	Physical segment name of segment updated.
Exit_Name	8	Character	8	Entry point name of exit routine.
Version	4	Character	2	XPCB version indicator. Along with release, identifies the level of the control block. Current version is V1.

Extended Segment Data Block (XSDB)

The XPCB points to the first XSDB. For path data, subsequent XSDBs are chained together. The XSDB points to the updated data that is passed to the exit routine. It contains additional information that is also passed. Table 8 and Table 9 describe the contents of the XSDB. For examples of how to code the XSDB, see “Sample Extended Segment Data Block (XSDB)” on page 52.

Table 8. XSDB By Offset

Offset Decimal	Field Name	Offset Decimal	Field Name	Offset Decimal	Field Name
0	Eyecatcher	4	Version	6	Release
8	Next_Ptr	12	Database_Name	20	Segment_Name
28	reserved	32	Segment_Level	34	Key_Length
36	Key_Ptr	40	LP_Key_Length	42	Segment_Length
44	Segment_Ptr	48	reserved		

Table 9. XSDB Alphabetically

Field Name	Offset Decimal	Data Type	Length Decimal	Field Description
Database_Name	12	Character	8	Physical database name for segment. Same as database name in XPCB.
Eyecatcher	0	Character	4	'XSDB'
Key_Length	34	Fixed	2	Length of key for segment. Zero if segment not keyed.
Key_Ptr	36	Pointer	4	Address of key for segment. Zero if segment not keyed.
LP_Key_Length	40	Fixed	2	Length of logical parent’s concatenated key included in segment data for logical children.
Next_Ptr	8	Pointer	4	Address of next XSDB in chain for path data. Zero for last XSDB in chain.
Release	6	Character	2	XSDB release indicator. Along with version, identifies level of control block. Current release is R1.
Segment_Ptr	44	Pointer	4	Address of physical segment data.
Segment_Length	42	Fixed	2	Length of physical segment data.
Segment_Level	32	Fixed	2	Level of segment in physical database.
Segment_Name	20	Character	8	Physical segment name for segment data passed in this block. Different from segment name in XPCB for path data.
Version	4	Character	2	XSDB version indicator. Along with release, identifies level of control block. Current version is V1.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of the XPCB address
13	Address of save area
14	Return address to IMS
15	Entry point of exit routine

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers. Return and reason codes are placed in the XPCB.

Return and Reason Codes

The XPCB contains fields for the exit routine to communicate its status to IMS. These fields are initialized to binary zeros. The return code allows the exit routine to define the type of condition encountered; the higher the number, the more severe the error. You can also assign a reason code to return codes of 8 or greater. This reason code allows you to identify the type and severity of the error. The reason code is for your use; IMS only uses the return code.

Table 10 outlines the return and reason codes that the exit routine returns and places in the XPCB. If the return code placed in the XPCB is invalid, an abend occurs and an invalid return code indicator is set.

Table 10. XPCB Return Codes

Return Code	Description	Action	DFS3314 Message
0	Good return.	Normal completion of exit routine.	No
4	Indicates the exit routine wants to ignore the DL/I call.	Exit routine is not called for any additional segments for this DL/I call.	No
8	Exit routine encountered an error during the DL/I call and wants to return to the application.	DL/I call is terminated without calling any other exit routines and control is returned to application program.	Yes
12	This copy of the exit routine is not to be called again. (Used with a “dummy” exit routine.)	Exit routine is deleted from storage.	Yes
16	Abend the exit routine and the application program.	Application program is abended with a U3314.	Yes
20	Updates to this region should not be captured nor should the exit routine be called.	Terminate data capture for this region.	Yes

Following an abend in an IFP or MPP region with return code 12 or 20, the interface control blocks are reinitialized and the exit work area is reset, enabling the exit routine to be called again.

Writing the Routine in Supported Languages

Although the Data Capture exit routine can be written in assembler, C, COBOL, or PL/I, you must follow certain guidelines depending upon which language you use. This section presents those guidelines.

Assembler

The exit routine is entered in primary mode, but the access registers can be nonzero.

C

C does not support variable-length character strings using integer lengths, such as those passed in the XPCB and XSDB. Key and segment data passed to the exit routine is terminated by “null” (binary zero) values. Any null value in the data itself might result in an invalid string length.

The following declarations and statements are used to locate the XPCB. Declare XPCB_TYPE_PTR as a pointer to the XPCB structure.

```
XPCB_TYPE_PTR *TPTR;
TPTR = (XPCB_TYPE_PTR *) __sysplist;
XPCB = *TPTR;
```

The exit routine must be defined as a main program with the PLIST(IMS) and ENV(IMS) options specified. Use the following format to specify these options:

```
#pragma runopt(env(IMS), plist(IMS))
```

COBOL

The exit routine operates under a separate run unit from the application program. The method used to establish the run unit depends on the RES/NORES compiler option. For resident (RES), the exit routine is given control via LINK. For nonresident (NORES), it is given control directly.

Recommendation:RES is recommended because the exit routine can be reentrant (RENT) and AMODE 31. With NORES, it must be AMODE 24 and it must not be reentrant.

You can use STOP RUN or GOBACK to terminate the exit routine run unit and return to the application program. You must use the NOSTAE and NOSPIE options if the exit routine makes any IMS calls.

The procedure division is:

```
exitname USING XPCB
```

PL/I

The exit routine must be compiled as a main program. The entry point must be PLICALLA, so that the exit routine can use the assembler interface.

The procedure statement is:

```
exitname:  PROCEDURE(XPCB_PTR) OPTIONS (MAIN);
```

Data Security and Integrity

The Data Capture exit routine is an extension of the application program with the same capabilities as the application program; the exit routine and the application have equal authorization and limitations. IMS and DB2 resources that the exit routine uses must be authorized in the application program’s IMS PSB or DB2 PLAN. This assures that the application program can access any IMS or DB2 data that is available to the exit routine.

The data and the exit routine operate in unprotected, key-8 storage. The exit routine is able to modify data or control blocks that can affect the successful operation of the application program. The data passed to the exit routine is the physical segment data. With PSB field sensitivity, this data might include data that is unavailable to the application.

Sample Extended Program Communication Block (XPCB)

This section provides examples of the XPCB in assembler, COBOL, and PL/I.

Assembler

```
         SPACE 3
XPCB     DSECT
XPCB_EYECATCHER         DS    CL4     "XPCB" EYECATCHER
XPCB_VERSION            DS    CL2     XPCB VERSION INDICATOR
XPCB_RELEASE            DS    CL2     XPCB RELEASE INDICATOR
XPCB_EXIT_NAME          DS    CL8     SEGMENT EXIT NAME
XPCB_EXIT_RETURN_CODE   DS    H       RETURN CODE FROM EXIT
XPCB_EXIT_REASON_CODE   DS    H       REASON CODE FROM EXIT
XPCB_DATABASE_NAME      DS    CL8     PHYSICAL DATA BASE NAME
XPCB_DBD_VERSION_PTR    DS    A       ADDRESS OF DBD VERSION ID
XPCB_SEGMENT_NAME       DS    CL8     PHYSICAL SEGMENT NAME
XPCB_CALL_FUNCTION      DS    CL4     CALL FUNCTION
XPCB_PHYSICAL_FUNCTION  DS    CL4     PHYSICAL CALL FUNCTION
                        DS    CL4
XPCB_DB_PCB_PTR         DS    A       ADDRESS OF DB PCB
XPCB_DB_PCB_NAME        DS    CL8     NAME OF DB PCB
XPCB_INQY_OUTPUT_PTR    DS    A       ADDRESS OF "INQY" OUTPUT
XPCB_IO_PCB_PTR         DS    A       ADDRESS OF I/O PCB
                        DS    H
XPCB_CONC_KEY_LENGTH    DS    H       LENGTH OF CONCATENATED KEY
XPCB_CONC_KEY_PTR       DS    A       ADDRESS OF CONCATENATED KEY
XPCB_DATA_XSDB_PTR      DS    A       ADDRESS OF XSDB FOR DATA
XPCB_BEFORE_XSDB_PTR    DS    A       ADDRESS OF XSDB FOR REPL DATA
XPCB_PATH_XSDB_PTR      DS    A       ADDRESS OF XSDB FOR PATH DATA
                        DS    F       RESERVED
                        DS    F       RESERVED
                        DS    F       RESERVED
XPCB_EXIT_WORK_PTR      DS    A       ADDRESS OF WORK AREA
XPCB_ZERO_POINTER       DS    A       ZERO ADDRESS
                        DS    F       RESERVED
XPCB_TIMESTAMP          DS    CL8     TIMESTAMP OF CALL
         EJECT
```

COBOL

```
01  XPCB.
    05  EYECATCHER                  PIC X(04).
    05  VERSION                     PIC X(02).
    05  RELEASE-ID                  PIC X(02).
    05  EXIT-NAME                   PIC X(08).
    05  EXIT-RETURN-CODE            PIC 9(04) COMP.
    05  EXIT-REASON-CODE            PIC 9(04) COMP.
    05  DATABASE-NAME               PIC X(08).
    05  DBD-VERSION-PTR             POINTER.
    05  SEGMENT-NAME                PIC X(08).
    05  CALL-FUNCTION               PIC X(04).
    05  PHYSICAL-FUNCTION           PIC X(04).
    05  FILLER                      PIC 9(08) COMP.
    05  DB-PCB-PTR                  POINTER.
    05  DB-PCB-NAME                 PIC X(08).
    05  INQY-OUTPUT-PTR             POINTER.
    05  IO-PCB-PTR                  POINTER.
    05  FILLER                      PIC 9(04) COMP.
    05  CONC-KEY-LENGTH             PIC 9(04) COMP.
    05  CONC-KEY-PTR                POINTER.
    05  DATA-XSDB-PTR               POINTER.
    05  BEFORE-XSDB-PTR             POINTER.
    05  PATH-XSDB-PTR               POINTER.
    05  FILLER                      POINTER.
    05  FILLER                      POINTER.
```

```
        05  FILLER                       POINTER.
        05  EXIT-WORK-PTR                POINTER.
        05  NULL-PTR                     POINTER.
        05  FILLER                       POINTER.
        05  TIMESTAMP                    PIC X(08).
```

PL/I

```
DECLARE
  1 XPCB              BASED(XPCB_PTR),
    3 EYECATCHER           CHAR(4),   /* "XPCB" EYECATCHER        */
    3 VERSION              CHAR(2),   /* XPCB VERSION INDICATOR   */
    3 RELEASE              CHAR(2),   /* XPCB RELEASE INDICATOR   */
    3 EXIT_NAME            CHAR(8),   /* SEGMENT EXIT NAME        */
    3 EXIT_RETURN_CODE     FIXED BINARY (15), /* RETURN CODE      */
    3 EXIT_REASON_CODE     FIXED BINARY (15), /* REASON CODE      */
    3 ATABASE_NAME        CHAR(8),   /* PHYSICAL DATA BASE NAME   */
    3 DBD_VERSION_PTR      POINTER,   /* ADDRESS OF DBD VERSION ID */
    3 SEGMENT_NAME         CHAR(8),   /* PHYSICAL SEGMENT NAME    */
    3 CALL_FUNCTION        CHAR(4),   /* CALL FUNCTION            */
    3 PHYSICAL_FUNCTION    CHAR(4),   /* DL/I PHYSICAL FUNCTION   */
    3 FILLER1              FIXED BINARY (31), /* RESERVED         */
    3 DB_PCB_PTR           POINTER,   /* ADDRESS OF DB PCB        */
    3 DB_PCB_NAME          CHAR(8),   /* NAME OF DB PCB           */
    3 INQY_OUTPUT_PTR      POINTER,   /* ADDRESS OF "INQY" OUTPUT */
    3 IO_PCB_PTR           POINTER,   /* ADDRESS OF I/O PCB       */
    3 FILLER2              FIXED BINARY (15), /* RESERVED         */
    3 CONC_KEY_LENGTH      FIXED BINARY (15), /* LENGTH OF FULLY  */
                                      /* CONCATENATED KEY FOR SEGM */
    3 CONC_KEY_PTR         POINTER,   /* ADDRESS OF PHYSICAL FULLY */
                                      /* CONCATENATED KEY FOR SEGM */
    3 DATA_XSDB_PTR        POINTER,   /* ADDRESS OF XSDB FOR      */
                                      /* PHYSICAL SEGMENT DATA    */
    3 BEFORE_XSDB_PTR      POINTER,   /* ADDRESS OF XSDB FOR      */
                                      /* PHYSICAL BEFORE DATA     */
    3 PATH_XSDB_PTR        POINTER,   /* ADDRESS OF XSDB FOR      */
                                      /* PHYSICAL PATH DATA       */
    3 FILLER3              POINTER,   /* RESERVED                 */
    3 FILLER4              POINTER,   /* RESERVED                 */
    3 FILLER5              POINTER,   /* RESERVED                 */
    3 EXIT_WORK_PTR        POINTER,   /* ADDRESS OF 256 BYTE AREA */
                                      /*       FOR THE EXIT       */
    3 NULL_PTR             POINTER,   /* NULL POINTER VALUE       */
    3 FILLER6              POINTER,   /* RESERVED                 */
    3 CALL_TIMESTAMP       CHAR(8),   /* TIMESTAMP OF CALL        */
    3 FILLER7              POINTER;   /* RESERVED FOR NULLS AT END */

DECLARE XPCB_PTR   POINTER;
```

Sample Extended Segment Data Block (XSDB)

This section provides examples of the XSDB in assembler, COBOL, and PL/I.

Assembler

```
         SPACE 3
XSDB     DSECT
XSDB_EYECATCHER         DS    CL4     "XSDB" EYECATCHER
XSDB_VERSION            DS    CL2     XSDB VERSION INDICATOR
XSDB_RELEASE            DS    CL2     XSDB RELEASE INDICATOR
XSDB_NEXT_PTR           DS    A       NEXT XSDB POINTER
XSDB_DATABASE_NAME      DS    CL8     PHYSICAL DATA BASE NAME
XSDB_SEGMENT_NAME       DS    CL8     PHYSICAL SEGMENT NAME
                        DS    CL4     RESERVED
XSDB_SEGMENT_LEVEL      DS    H       SEGMENT DATA BASE LEVEL
XSDB_KEY_LENGTH         DS    H       LENGTH OF PHYSICAL KEY
```

```
XSDB_KEY_PTR            DS    A       ADDRESS OF PHYSICAL KEY
XSDB_LP_KEY_LENGTH      DS    H       LENGTH OF LOGICAL PARENT KEY
XSDB_SEGMENT_LENGTH     DS    H       LENGTH OF SEGMENT DATA
XSDB_SEGMENT_DATA_PTR   DS    A       ADDRESS OF SEGMENT DATA
                        DS    F       RESERVED
                        DS    F       RESERVED
```

COBOL

```
01  XSDB
    05  EYECATCHER                    PIC X(4).
    05  VERSION                       PIC X(2).
    05  RELEASE-ID                    PIC X(2).
    05  NEXT-PTR                      POINTER.
    05  DATABASE-NAME                 PIC X(8).
    05  SEGMENT-NAME                  PIC X(8).
    05  FILLER                        PIC X(4).
    05  SEGMENT-LEVEL                 PIC 9(4) COMP.
    05  KEY-LENGTH                    PIC 9(4) COMP.
    05  KEY-PTR                       POINTER.
    05  LP-KEY-LENGTH                 PIC 9(4) COMP.
    05  SEGMENT-LENGTH                PIC 9(4) COMP.
    05  SEGMENT-DATA-PTR              POINTER.
    05  FILLER                        POINTER.
    05  FILLER                        POINTER.
```

PL/I

```
DECLARE
  1 XSDB              BASED(XSDB_PTR),
    3 EYECATCHER            CHAR(4),    /* "XSDB" EYECATCHER         */
    3 VERSION               CHAR(2),    /* XSDB VERSION INDICATOR    */
    3 RELEASE               CHAR(2),    /* XSDB RELEASE INDICATOR    */
    3 NEXT_PTR              POINTER,    /* NEXT XSDB POINTER         */
    3 DATABASE_NAME         CHAR(8),    /* PHYSICAL DATA BASE NAME   */
    3 SEGMENT_NAME          CHAR(8),    /* PHYSICAL SEGMENT NAME     */
    3 FILLER1               CHAR(4),    /* RESERVED                  */
    3 SEGMENT_LEVEL FIXED BINARY (15),  /* SEGMENT DATA BASE LEVEL   */
    3 KEY_LENGTH    FIXED BINARY (15),  /* LENGTH OF PHYSICAL KEY    */
    3 KEY_PTR               POINTER,    /* ADDRESS OF PHYSICAL KEY   */
    3 LP_KEY_LENGTH FIXED BINARY (15),  /* RESERVED                  */
    3 SEGMENT_LENGTH FIXED BINARY (15), /* LENGTH OF SEGMENT DATA    */
    3 SEGMENT_DATA_PTR      POINTER,    /* ADDRESS OF SEGMENT DATA   */
    3 FILLER3               POINTER,    /* RESERVED                  */
    3 FILLER4               POINTER,    /* RESERVED                  */
    3 FILLER5               POINTER;    /* RESERVED FOR NULLS AT END */

DECLARE XSDB_PTR            POINTER;
```

Sample Data Capture Exit Routine

This section provides examples of the Data Capture exit routine in COBOL and PL/I, respectively. The exit routine can also be written in assembler or C.

COBOL

```
        IDENTIFICATION DIVISION.
          PROGRAM-ID. DLICDCE.
       *---------------------------------------------------------------*
       *REMARKS.                                                        *
       *---------------------------------------------------------------*
       *    DESCRIPTIVE NAME : HOSPITAL DATA BASE SEGMENT EXIT         *
       *---------------------------------------------------------------*
       *    THIS IS A SAMPLE IMS EXIT. THIS WILL BE CALLED BY IMS.     *
       *    THIS PROGRAM PROPAGATES DATA FROM IMS TO DB2 SYNCHRONOUSLY.*
```

```
*    THE NAME OF THIS PROGRAM LOAD MODULE IS SPECIFIED          *
*    ON SEGM MACRO DURING DBDGEN FOR THE HOSPITAL DATA BASE.    *
*                                                               *
*    THE DATA OPTIONS SELECTED FOR THIS EXIT :                  *
*    EXIT=(KEY,DATA,NOPATH,CASCADE)                             *
*---------------------------------------------------------------*
*    INPUT FOR THIS PROGRAM : XPCB, XSDB.                       *
*                                                               *
*    OUTPUT:  DISPLAY A MESSAGE WHEN THE IMS UPDATE IS NOT      *
*             ISRT, REPL, DELE, CASC. DISPLAY 'SQLERRM' WHEN    *
*             SQLERROR OCCURS.                                  *
*                                                               *
*    UPDATES: UPDATES DB2 ILLNESS TABLE                         *
*---------------------------------------------------------------*
*    LOGIC:   THIS PROGRAM IS CALLED BY IMS AFTER THE IMS UPDATE*
*             TO ILLNESS SEGMENT AND BEFORE IMS RETURNS TO THE  *
*             IMS APPLICATION PROGRAM.                          *
*                                                               *
*             XPCB IS RECEIVED AS INPUT TO THIS PROGRAM.        *
*             IF THERE IS NO ADDRESS OF XSDB IN XPCB THIS       *
*             PROGRAM WILL RETURNS TO IMS OTHERWISE -           *
*                                                               *
*    LOGIC:   THIS PROGRAM IS CALLED BY IMS AFTER THE IMS UPDATE*
*             WE GET THE ADDRESS OF XSDB FROM XPCB, FROM XSDB   *
*             WE GET THE ADDRESS OF ILLNESS SEGMENT CONCATENATED*
*             KEY, AND ADDRESS OF THE PHYSICAL SEGMENT DATA     *
*                                                               *
*             UPDATE THE DB2 ILLNESS TABLE WITH THE UPDATED IMS *
*             SEGMENT DATA.                                     *
* --------------------------------------------------------------*
 INSTALLATION.  IBM - SANTA TERESA LABORATORY.
 DATE-WRITTEN.  JANUARY 1990.
 ENVIRONMENT DIVISION.

 CONFIGURATION SECTION.
 SOURCE-COMPUTER. IBM-3090.
 OBJECT-COMPUTER. IBM-3090.
 DATA DIVISION.

 WORKING-STORAGE SECTION.
      EXEC SQL
        INCLUDE SQLCA
      END-EXEC.
*--- DB2 ILLNESS TABLE DECLARATION

      EXEC SQL
        DECLARE SYSADM.ILLNESS TABLE
          (ILLDATE VARCHAR (6)    NOT NULL,
           PATNO   VARCHAR (5)    NOT NULL,
           ILLNAME VARCHAR (10)   NOT NULL)
      END-EXEC.
*---

 01  W-POINTER                       POINTER.
 01  W-POINTER-R REDEFINES W-POINTER PIC 9(8) COMP.
 LINKAGE SECTION.
*--- EXIT SEGMENT CONTROL BLOCK

 01  XPCB.
     05  EYECATCHER                    PIC X(04).
     05  VERSION                       PIC X(02).
     05  RELEASE-ID                    PIC X(02).
     05  EXIT-NAME                     PIC X(08).
     05  EXIT-RETURN-CODE              PIC 9(04) COMP.
     05  EXIT-REASON-CODE              PIC 9(04) COMP.
     05  DATABASE-NAME                 PIC X(08).
     05  DBD-VERSION-PTR               POINTER.
     05  SEGMENT-NAME                  PIC X(08).
     05  CALL-FUNCTION                 PIC X(04).
     05  PHYSICAL-FUNCTION             PIC X(04).
     05  FILLER                        PIC 9(08) COMP.
     05  DB-PCB-PTR                    POINTER.
     05  DB-PCB-NAME                   PIC X(08).
     05  INQY-OUTPUT-PTR               POINTER.
     05  IO-PCB-PTR                    POINTER.
     05  FILLER                        PIC 9(04) COMP.
     05  CONC-KEY-LENGTH               PIC 9(04) COMP.
     05  CONC-KEY-PTR                  POINTER.
     05  DATA-XSDB-PTR                 POINTER.
     05  BEFORE-XSDB-PTR               POINTER.
     05  PATH-XSDB-PTR                 POINTER.
     05  FILLER                        POINTER.
     05  FILLER                        POINTER.
     05  FILLER                        POINTER.
     05  EXIT-WORK-PTR                 POINTER.
     05  NULL-PTR                      POINTER.
     05  FILLER                        POINTER.
     05  TIMESTAMP                     PIC X(08).
*--- EXIT SEGMENT DATA BLOCK

 01  DATA-XSDB.
     05  EYECATCHER                    PIC X(4).
     05  VERSION                       PIC X(2).
     05  RELEASE-ID                    PIC X(2).
     05  NEXT-PTR                      POINTER.
     05  DATABASE-NAME                 PIC X(8).
     05  SEGMENT-NAME                  PIC X(8).
     05  FILLER                        PIC X(4).
     05  SEGMENT-LEVEL                 PIC 9(4) COMP.
     05  KEY-LENGTH                    PIC 9(4) COMP.
     05  KEY-PTR                       POINTER.
     05  FILLER                        PIC 9(4) COMP.
     05  SEGMENT-LENGTH                PIC 9(4) COMP.
     05  SEGMENT-DATA-PTR              POINTER.
     05  FILLER                        POINTER.
     05  FILLER                        POINTER.
*--- ILLNESS SEGMENT DATA

 01  LS-SEGMENT.

     03  LS-ILLDATE                    PIC X(6).
     03  LS-ILLNAME                    PIC X(10).
*--- ILLNESS SEGMENT CONCATENATED KEY

 01  XPCB-CONCKEY.

     02  LS-PATNO                      PIC X(5).
     02  LS-ILLDT                      PIC X(6).
 PROCEDURE DIVISION USING XPCB.

     SET W-POINTER TO DATA-XSDB-PTR.
*--- LENGTH ZERO IF NOT CAPTURED

*    IF W-POINTER-R EQUAL ZEROES GOBACK
*       GOBACK
*    END-IF
*----

     SET  ADDRESS OF DATA-XSDB TO DATA-XSDB-PTR.
     SET  ADDRESS OF XPCB-CONCKEY TO CONC-KEY-PTR.
     SET  ADDRESS OF LS-SEGMENT TO SEGMENT-DATA-PTR.
*----
       EXEC SQL
         WHENEVER SQLWARNING CONTINUE
```

```
                   END-EXEC
                   EXEC SQL
                     WHENEVER SQLERROR GO TO BADSQL
                   END-EXEC
                   EXEC SQL
                     WHENEVER NOT FOUND GO TO BADSQL
                   END-EXEC
           *----

                     IF PHYSICAL-FUNCTION OF XPCB = "ISRT"

                        EXEC SQL
                          INSERT INTO SYSADM.ILLNESS
                          VALUES (::LS-ILLDATE,::LS-PATNO,
                          ::LS-ILLNAME)
                        END-EXEC
                     ELSE
                       IF PHYSICAL-FUNCTION OF XPCB = "CASC" OR
                          PHYSICAL-FUNCTION OF XPCB = "DLET"
                          EXEC SQL
                            DELETE FROM SYSADM.ILLNESS
                            WHERE (PATNO = ::LS-PATNO AND
                                   ILLDATE = ::LS-ILLDATE)
                          END-EXEC
                       ELSE

                         IF PHYSICAL-FUNCTION OF XPCB = "REPL"

                            EXEC SQL
                              UPDATE SYSADM.ILLNESS
                              SET ILLNAME = ::LS-ILLNAME
                              WHERE (ILLDATE = ::LS-ILLDATE AND
                                     PATNO = ::LS-PATNO)
                            END-EXEC
                         ELSE

                           DISPLAY "FUNCTION WASNT ISRT, REPL, DLET, OR CASC"
                           DISPLAY "--- NO SQL ACTION WAS TAKEN"
                           DISPLAY "PHYS FUNCTION IS "
                           DISPLAY PHYSICAL-FUNCTION OF XPCB
                         END-IF
                       END-IF
                     END-IF.
                     DISPLAY "SQLCODE " SQLCODE.
                     STOP RUN.
             BADSQL.

                 DISPLAY "SQLERRM".
                 MOVE 8        TO EXIT-RETURN-CODE OF XPCB.
                 MOVE SQLCODE  TO EXIT-REASON-CODE OF XPCB.
                 STOP RUN.
```

PL/I

```
DLI2DB2: PROCEDURE(XPCB_PTR) OPTIONS(MAIN);
/*
 *---------------------------------------------------------------*
 *REMARKS.                                                        *
 *---------------------------------------------------------------*
 *    DESCRIPTIVE NAME : HOSPITAL DATA BASE SEGMENT EXIT         *
 *---------------------------------------------------------------*
 *    THIS IS A SAMPLE IMS EXIT THAT WILL BE CALLED BY IMS.      *
 *    THIS PROGRAM PROPAGATES DATA FROM IMS TO DB2 SYNCHRONOUSLY.*
 *    THE NAME OF THIS PROGRAM LOAD MODULE IS SPECIFIED          *
 *    ON SEGM MACRO DURING DBDGEN FOR THE HOSPITAL DATA BASE.    *
 *                                                                *
 *    THE DATA OPTIONS SELECTED FOR THIS EXIT ARE:              *
 *    EXIT=(DLI2DB2,PATH,DATA,(CASCADE,PATH,DATA,NOKEY)         *
 *--------------------------------------------------------------*
 *                                                              *
 *    INPUT FOR THIS PROGRAM : XPCB, XSDB.                      *
 *                                                              *
 *    OUTPUT:  DISPLAY 'SQLERRM' WHEN SQLERROR OCCURS.          *
 *    UPDATES: UPDATES DB2 TREATMT TABLE                        *
 *                                                              *
 *           : RETURNS REASON CODE 14 RETURN CODE 16 WHEN PATH  *
 *             NOT SPECIFIED ON THE DBDGEN EXIT STATEMENT,      *
 *             RESULTING IN AN ABEND U3314.                     *
 *                                                              *
 *--------------------------------------------------------------*
 *    LOGIC:   THIS PROGRAM IS CALLED BY IMS AFTER AN UPDATE TO  *
 *             THE TREATMT SEGMENT AND BEFORE IMS RETURNS TO     *
 *             IMS APPLICATION PROGRAM.                         *
 *                                                              *
 *             THE ADDRESS OF AN XPCB IS PASSED TO THIS PROGRAM  *
 *             FROM IMS. THE XPCB WILL PROVIDE THE ADDRESSES OF  *
 *             THE XSDB FOR DATA, PATH DATA AND BEFORE DATA.     *
 *                                                              *
 *             UPDATE THE DB2 TREATMT TABLE WITH THE UPDATED IMS *
 *             SEGMENT DATA.                                    *
 *                                                              *
 *   HOSPITAL   ***********                                     *
 *   DATA BASE  *         *                                     *
 *              * PATIENT *  KEY FIELD IS PATNO                 *
 *              *         *                                     *
 *              ***********                                     *
 *                   *                                          *
 *                   *                                          *
 *              ***********                                     *
 *              *         *                                     *
 *              * ILLNESS *  KEY FIELD IS ILLDATE               *
 *              *         *                                     *
 *              ***********                                     *
 *                   *                                          *
 *                   *                                          *
 *              ***********  KEY FIELD IS TRTDATE               *
 *              *         *  FIELD, MEDICINE                    *
 *              * TREATMT *  FIELD, QUANTITY                    *
 *              *         *  FIELD, DOCTOR (NOT IN DB2 TABLE)   *
 *              ***********                                     *
 *                                                              *
 *                                                              *
 *      TREATMENT TABLE                                         *
 *                                                              *
 *      ***************************************************     *
 *      * PATNUMB * DATEILL * DATETRT * MEDICAT * AMOUNT  *     *
 *      ***************************************************     *
 *                                                              *
 * -------------------------------------------------------------*
 */
/* ************************************************************  */
/*                                                               */
/*      E X T E N D E D   D A T A  B A S E  P C B  -- X P C B    */
/*                                                               */
/* ************************************************************* */

DECLARE
```

```
  1 XPCB               BASED(XPCB_PTR),
    3 EYECATCHER            CHAR(4),   /* "XPCB" EYECATCHER         */
    3 VERSION               CHAR(2),   /* XPCB VERSION INDICATOR    */
    3 RELEASE               CHAR(2),   /* XPCB RELEASE INDICATOR    */
    3 EXIT_NAME             CHAR(8),   /* SEGMENT EXIT NAME         */
    3 EXIT_RETURN_CODE      FIXED BINARY (15), /* RETURN CODE       */
    3 EXIT_REASON_CODE      FIXED BINARY (15), /* REASON CODE       */
    3 DATABASE_NAME         CHAR(8),   /* PHYSICAL DATA BASE NAME   */
    3 DBD_VERSION_PTR       POINTER,   /* ADDRESS OF DBD VERSION ID */
    3 SEGMENT_NAME          CHAR(8),   /* PHYSICAL SEGMENT NAME     */
    3 CALL_FUNCTION         CHAR(4),   /* CALL FUNCTION             */
    3 PHYSICAL_FUNCTION     CHAR(4),   /* DL/I PHYSICAL FUNCTION    */
    3 FILLER1               FIXED BINARY (31), /* RESERVED          */
    3 DB_PCB_PTR            POINTER,   /* ADDRESS OF DB PCB         */
    3 DB_PCB_NAME           CHAR(8),   /* NAME OF DB PCB            */
    3 INQY_OUTPUT_PTR       POINTER,   /* ADDRESS OF "INQY" OUTPUT  */
    3 IO_PCB_PTR            POINTER,   /* ADDRESS OF I/O PCB        */
    3 FILLER2               FIXED BINARY (15), /* RESERVED          */
    3 CONC_KEY_LENGTH       FIXED BINARY (15), /* LENGTH OF FULLY   */
                                       /* CONCATENATED KEY FOR SEGM */
    3 CONC_KEY_PTR          POINTER,   /* ADDRESS OF PHYSICAL FULLY */
                                       /* CONCATENATED KEY FOR SEGM */
    3 DATA_XSDB_PTR         POINTER,   /* ADDRESS OF XSDB FOR       */
                                       /* PHYSICAL SEGMENT DATA     */
    3 BEFORE_XSDB_PTR       POINTER,   /* ADDRESS OF XSDB FOR       */
                                       /* PHYSICAL BEFORE DATA      */
    3 PATH_XSDB_PTR         POINTER,   /* ADDRESS OF XSDB FOR       */
                                       /* PHYSICAL PATH DATA        */
    3 FILLER3               POINTER,   /* RESERVED                  */
    3 FILLER4               POINTER,   /* RESERVED                  */
    3 FILLER5               POINTER,   /* RESERVED                  */
    3 EXIT_WORK_PTR         POINTER,   /* ADDRESS OF 256 BYTE AREA  */
                                       /*       FOR THE EXIT        */
    3 NULL_PTR              POINTER,   /* NULL POINTER VALUE        */
    3 FILLER6               POINTER,   /* RESERVED                  */
    3 CALL_TIMESTAMP        CHAR(8),   /* TIMESTAMP OF CALL         */
    3 FILLER7               POINTER;   /* RESERVED FOR NULLS AT END */
DECLARE XPCB_PTR   POINTER;
/* ************************************************************** */
/*                                                                */
/*      E X T E N D E D   S E G M E N T  D A T A   -- X S D B     */
/*                                                                */
/* ************************************************************** */

DECLARE
  1 XSDB              BASED(XSDB_PTR),
    3 EYECATCHER            CHAR(4),    /* "XSDB" EYECATCHER         */
    3 VERSION               CHAR(2),    /* XSDB VERSION INDICATOR    */
    3 RELEASE               CHAR(2),    /* XSDB RELEASE INDICATOR    */
    3 NEXT_PTR              POINTER,    /* NEXT XSDB POINTER         */
    3 DATABASE_NAME         CHAR(8),    /* PHYSICAL DATA BASE NAME   */
    3 SEGMENT_NAME          CHAR(8),    /* PHYSICAL SEGMENT NAME     */
    3 FILLER1               CHAR(4),    /* RESERVED                  */
    3 SEGMENT_LEVEL FIXED BINARY (15),  /* SEGMENT DATA BASE LEVEL   */
    3 KEY_LENGTH    FIXED BINARY (15),  /* LENGTH OF PHYSICAL KEY    */
    3 KEY_PTR               POINTER,    /* ADDRESS OF PHYSICAL KEY   */
    3 FILLER2       FIXED BINARY (15),  /* RESERVED                  */
    3 SEGMENT_LENGTH FIXED BINARY (15), /* LENGTH OF SEGMENT DATA    */
    3 SEGMENT_DATA_PTR      POINTER,    /* ADDRESS OF SEGMENT DATA   */
    3 FILLER3               POINTER,    /* RESERVED                  */
    3 FILLER4               POINTER,    /* RESERVED                  */
    3 FILLER5              POINTER;    /* RESERVED FOR NULLS AT END */

DECLARE XSDB_PTR           POINTER;
DECLARE
  1 SEGMENT_XSDB    LIKE XSDB  BASED(XPCB.DATA_XSDB_PTR);
DECLARE                /* TREATMENT SEGMENT    */
  1 SEGMENT_DATA   BASED(SEGMENT_XSDB.SEGMENT_DATA_PTR),
    3 SEGMENT_DATA_TRTDATE   CHAR(6),      /* SEGMENT KEY */
    3 SEGMENT_DATA_MEDICINE  CHAR(10),
    3 SEGMENT_DATA_QUANTITY  CHAR(4),
    3 SEGMENT_DATA_DOCTOR    CHAR(10);
DECLARE
  1 BEFORE_XSDB  LIKE XSDB  BASED(XPCB.BEFORE_XSDB_PTR);
DECLARE                /* BEFORE TREATMENT SEGMENT */
  1 BEFORE_DATA  BASED(BEFORE_XSDB.SEGMENT_DATA_PTR),
    3 BEFORE_DATA_TRTDATE   CHAR(6),      /* SEGMENT KEY */
    3 BEFORE_DATA_MEDICINE  CHAR(10),
    3 BEFORE_DATA_QUANTITY  CHAR(4),
    3 BEFORE_DATA_DOCTOR    CHAR(10);
DECLARE
  1 PATH_XSDB    LIKE XSDB  BASED(PATH_XSDB_PTR);
DECLARE                /*  PATIENT  SEGMENT    */
  1 PATH_DATA BASED(PATH_XSDB.SEGMENT_DATA_PTR),
    3 PATHSEG_PATNO          CHAR(5),      /* SEGMENT KEY */
    3 PATHSEG_NAME           CHAR(10),
    3 PATHSEG_ADDR           CHAR(30);
DECLARE
  1 PATH2_XSDB    LIKE XSDB  BASED(PATH2_XSDB_PTR);
DECLARE                /*  PATIENT  SEGMENT    */
  1 PATH2_DATA BASED(PATH2_XSDB.SEGMENT_DATA_PTR),
    3 PATH2SEG_ILLDATE        CHAR(6),      /* SEGMENT KEY */
    3 PATH2SEG_ILLNAME        CHAR(10);
DECLARE PATH2_XSDB_PTR            POINTER;
DECLARE                /* TREATMENT TABLE ROW  */
  1 TREATROW    BASED(XPCB.EXIT_WORK_PTR),
    3 COL_PATNUM         CHAR(5),      /* FROM LEVEL 1 KEY */
    3 COL_ILLDATE        CHAR(6),      /* FROM LEVEL 2 KEY */
    3 COL_TRTDATE        CHAR(6),      /* FROM LEVEL 3 KEY */
    3 COL_MEDICINE       CHAR(10),      /* FROM LEVEL 3 */
    3 COL_QUANTITY       CHAR(4);      /* FROM LEVEL 3 */
           EXEC SQL
             INCLUDE SQLCA;
     /* - DB2 TREATMENT TABLE DECLARATION  */

           EXEC SQL
             DECLARE SYSADM.TREATMNT TABLE
               (PATNUMB VARCHAR (5)    NOT NULL,
                DATEILL VARCHAR (6)    NOT NULL,
                DATETRT VARCHAR (6)    NOT NULL,
                MEDICAT VARCHAR (10)   NOT NULL,
                AMOUNT  VARCHAR (4)    NOT NULL);

DECLARE                                 /* CALL FUNCTIONS */
    INSERT_FUNCTION       CHAR(4) STATIC INIT('ISRT'),
    DELETE_FUNCTION       CHAR(4) STATIC INIT('DLET'),
    REPLACE_FUNCTION      CHAR(4) STATIC INIT('REPL'),
    CASCADE_FUNCTION      CHAR(4) STATIC INIT('CASC');
DECLARE ZERO      FIXED BINARY (31) STATIC
                   INIT(0);
DECLARE SIXTEEN   FIXED BINARY (31) STATIC
                   INIT(16);
```

```
PATH2_XSDB_PTR = PATH_XSDB.NEXT_PTR;
TREATROW.COL_PATNUM = PATH_DATA.PATHSEG_PATNO;
TREATROW.COL_ILLDATE = PATH2_DATA.PATH2SEG_ILLDATE;
TREATROW.COL_TRTDATE = SEGMENT_DATA.SEGMENT_DATA_TRTDATE;
TREATROW.COL_MEDICINE = SEGMENT_DATA.SEGMENT_DATA_MEDICINE;
TREATROW.COL_QUANTITY = SEGMENT_DATA.SEGMENT_DATA_QUANTITY;

 EXEC SQL
      WHENEVER SQLWARNING CONTINUE;
 EXEC SQL
      WHENEVER SQLERROR GOTO BADSQL;
 EXEC SQL
      WHENEVER NOT FOUND GOTO BADSQL;
IF XPCB.PATH_XSDB_PTR = XPCB.NULL_PTR
THEN DO;
GOTO BADPATH;   /* PATH NOT SPECIFIED */
END;
ELSE DO;       /* PRE-SET CODES TO ZERO */
XPCB.EXIT_RETURN_CODE = ZERO;
XPCB.EXIT_REASON_CODE = ZERO;
END;
     /*====================================*/
     /* IF CALLED FOR DELETE OR CASCADE,   */
     /* PERFORM THE DB2 DELETE.            */
     /*====================================*/
IF XPCB.PHYSICAL_FUNCTION = DELETE_FUNCTION
THEN DO;

   EXEC SQL
     DELETE FROM SYSADM.TREATMNT
     WHERE PATNUMB = ::TREATROW.COL_PATNUM AND
     DATEILL = ::TREATROW.COL_ILLDATE AND
          DATETRT = ::TREATROW.COL_TRTDATE;
END;
     /*==========================================*/
     /* IF CALLED FOR INSERT, DO DB2 INSERT CALL */
     /*==========================================*/
IF XPCB.CALL_FUNCTION = INSERT_FUNCTION
THEN DO;
  EXEC SQL
     INSERT INTO SYSADM.TREATMNT
     VALUES(::TREATROW.COL_PATNUM,
            ::TREATROW.COL_ILLDATE,
            ::TREATROW.COL_TRTDATE,
            ::TREATROW.COL_MEDICINE,
            ::TREATROW.COL_QUANTITY);
END;
     /*=====================================*/
     /* IF CALLED FOR REPLACE, UPDATE THE   */
     /* THE DB2 ROW, IF A FIELD DESTINED TO */
     /* THE DB2 DATA BASE HAS BEEN CHANGED. */
     /*=====================================*/
IF XPCB.CALL_FUNCTION = REPLACE_FUNCTION
THEN DO;    /* REPLACE */
 IF (SEGMENT_DATA.SEGMENT_DATA_MEDICINE &#170;=
 BEFORE_DATA.BEFORE_DATA_MEDICINE) |
 (SEGMENT_DATA.SEGMENT_DATA_QUANTITY &#170;=
 BEFORE_DATA.BEFORE_DATA_QUANTITY)
 THEN DO;   /* UPDATE */
   EXEC SQL
```

```
      UPDATE SYSADM.TREATMNT
      SET  MEDICAT = ::SEGMENT_DATA.SEGMENT_DATA_MEDICINE,
           AMOUNT  = ::SEGMENT_DATA.SEGMENT_DATA_QUANTITY
      WHERE PATNUMB = ::TREATROW.COL_PATNUM AND
      DATEILL = ::TREATROW.COL_ILLDATE AND
      DATETRT = ::TREATROW.COL_TRTDATE;
    END;       /* OF UPDATE  */
   END;        /* OF REPLACE */
STOP;
   BADSQL: DO;
   DISPLAY(SQLERRM);
   XPCB.EXIT_RETURN_CODE = 16;
   XPCB.EXIT_REASON_CODE = SQLCODE;
   END;
   BADPATH: DO;
   XPCB.EXIT_RETURN_CODE = 16;
   XPCB.EXIT_REASON_CODE = 14;
   END;
END DLI2DB2B;
```

Chapter 3. Data Conversion User Exit Routine (DFSDBUX1)

This section documents General-Use Programming Interface and Associated Guidance Information. See “Notices” on page xxi to understand this classification of information.

The Data Conversion user exit routine (DFSDBUX1) gets control at the beginning of a DL/I call and at the end of the call. In the exit routine, you can modify segment search arguments, the key feedback area, the I/O area, and the status code.

In this Chapter:

About This Routine

The purpose of the Data Conversion exit routine (DFSDBUX1) is to provide a method for modifying segment search arguments, the key feedback area, the I/O area, and the status code.

Attributes of the Routine

Regardless of its function, the exit routine must be written in assembler language, C language, COBOL, or PL/I. Routines written in high- level languages running under Language Environment for OS/390 are not supported.

Link-Editing the Routine

Link-edit the exit routine DFSDBUX1 with the RENT attribute into an APF-authorized library. This library can be either IMS.RESLIB, SYS1.LINKLIB, or any partitioned data set that can be accessed by a JOBLIB or a STEPLIB DD statement for the IMS control, SAS, batch, or CICS region.

Loading the Routine

IMS attempts to load the exit routine upon the first database call. If the exit routine fails to load, IMS does not attempt to load it again.

The exit routine is called if the following two conditions are both satisfied:

- The DL/I data conversion code is installed (APAR PQ13662).
- DFSDBUX1 exists in IMS.RESLIB.

Other Considerations

After the DL/I conversion code is installed, a DBDGEN is **not** required. However, it is recommended that you perform a DBDGEN with the DATXEXIT=YES parameter for DBDs that require the exit routine.

If you do **not** specify DATXEXIT=YES for a DBD, the call analyzer (DFSDLA00) issues a DFS2097I message if the exit routine specifies that it should continue to be called for that DBD. After issuing the DFS2097I message, DFSDLA00 dynamically sets the DATXEXIT=YES indicator for the DBD and continues calling the exit routine. Thus, the DFS2097I message appears only once per DBD.

If you link-edit an exit routine and want to prevent it from being called, you must remove DFSDBUX1 from the library where you edited it.

If DFSDBUX1 is available to IMS, it is called regardless of what is specified by the DATXEXIT parameter. If the exit routine determines that the exit routine should not be called again for the DBD, the routine should return X'FF' in the SRCHFLAG field in the JCB (SRCHFLAG EQUA JCBWKR55). X'FF' causes DFSDLA00 to dynamically mark the DBD as not requiring the exit routine. In this case, the exit routine is not called again for that DBD for the duration of the execution of this IMS **or** until the DMB is purged from the DMB pool.

You can also control which databases DFSDBUX1 acts upon, once called, by adding the database names to a table that the exit can check.

Preloading the exit routine is not necessary. Once loaded, the exit routine remains loaded until region termination.

Using IMS Callable Services with This Routine

This exit is not eligible to use IMS Callable Services.

Issuing SVC Calls

In an online environment, the exit routine might be running in cross-memory mode. To prevent 0F8 abends, the exit should avoid issuing SVC calls.

Communicating with IMS

IMS uses the general purpose registers and several IMS control blocks to communicate with the DFSDBUX1 exit routine.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
0	The characters 'IN' at the start of the DL/I call and the characters 'OUT' at the end of the DL/I call.
1	Address of the Partition Specification Table.
3	Address of the Database Program Control Block (DBPCB).
5	Address of the PSB Directory (PDIR).
6	Address of the System Contents Directory (SCD).
7	Address of the Program Specification Block (PCB).
9	Address of the Job Control Block (JCB).
10	Address of the Segment Descriptor Block (SDB).
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of exit routine.

Contents of Register on Exit

Before returning to IMS, the exit routine must restore registers 0 through 14. Register 15 must be set as follows:

Register	Contents
0	The exit routine has successfully processed the request.
non-0	The exit routine has set a status code or pseudoabend.

Data Security and Integrity

The exit routine is an extension of the application program with the same capabilities as the application program; the exit routine and the application have equal authorization and limitations.

In batch, the data and the exit routine operate in unprotected key-8 storage. Online, the data and the exit routine operate in unprotected key-7 storage. The exit routine is able to modify data or control blocks that can affect the successful operation of the application program.

Chapter 4. Data Entry Database Randomizing Routine (DBFHDC40/ DBFHDC44)

This chapter describes the Data Entry Database (DEDB) Randomizing routine. The chapter specifies the attributes of the routine and how the routine communicates with IMS.

In this Chapter:

About This Routine

A DEDB randomizing module is required for placing root segments in or retrieving them from a DEDB. Several DEDBs can share the same routine, but all AREAs in a DEDB must use the same routine.

If you are using data sharing, you must use the same randomizing routine on both systems.

IMS supplies sample DEDB randomizing modules (DBFHDC40 and DBFHDC44) on IMS.DBSOURCE. You can use one of these IMS-supplied routines or you can write your own.

A randomizing module uses a mathematical technique to convert a key into an address. The same key always converts to the same address. The randomizing module required by IMS must convert a key field value into a relative anchor point number. A randomizing routine results in a relative anchor point number that can range from 0 to $2^{31}-1$.

The key field value is supplied by an application program in the data itself for inserting segments into the database, and in an SSA (segment search argument) for retrieving segments from a database.

Related Reading: For general guidelines on writing DL/I exit routines, see "Chapter 1. Guidelines for Writing IMS Exit Routines" on page 3 of this book.

This routine must be written and link-edited as *reentrant* (RENT) like the one that IMS supplies. The routine receives control and must return control in 31-bit addressing mode (AMODE 31). It must be able to execute in cross-memory and TASK modes.

You must reassemble the modules you wrote for use with previous IMS releases because of changes to the control blocks.

Table 11 shows the attributes of the Data Entry Database Randomizing routine.

Table 11. Data Entry Database Randomizing Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.

Table 11. Data Entry Database Randomizing Routine Attributes (continued)

Attribute	Description
Naming convention	The name given to the load module used for randomizing functions with a specific database should also appear in the DBD generation associated with the database. The load module name must be the value of the “mod” parameter of the RMNAME= operand on the DBD statement in the DEDB DBD generation. **Related Reading:**For details on coding this parameter, see the chapter on “DBD Generation” in *IMS/ESA Utilities Reference: System*.
Link editing	Follow the guidelines described in “Link-Editing the Routines” on page 5. After you compile and test your randomizing module, link-edit it into IMS.RESLIB, SYS1.LINKLIB, or any operating system partitioned data set that can be accessed by a JOBLIB or STEPLIB JCL statement for the IMS control and SAS regions.
Including the routine	No special steps are needed to include this routine.
IMS callable services	This exit is not eligible to use IMS Callable Services.
Sample routine location	IMS.DBSOURCE (member name DBFHDC40). For more information, see “Using the Sample DEDB Randomizing Routines (DBFHDC40)” on page 70.

Loading the Routine

All randomizing modules are loaded from their resident library by IMS. The name of the module is the name you specified in the RMNAME parameter of the DBD statement of the database description (DBD).

Related Reading: For details on coding the RMNAME parameter, see *IMS/ESA Utilities Reference: Database Manager*.

The necessary randomizing module associated with a specific database is brought into main storage at initialization time. Though not recommended, the randomizing module can also be placed in the LPA (linkpack area). This allows one copy of the module to service several databases that are concurrently open. When running under MVS, the randomizing module is loaded into the Common Service Area (CSA). If you link-edited with RMODE ANY, you can load it into the Extended Common Service Area (ECSA).

Related Reading: For more information, see “Chapter 1. Guidelines for Writing IMS Exit Routines” on page 3.

Activating the Routine

When an application program issues a Get Unique or Insert call that operates on a root segment of a DEDB database, the user-supplied randomizing module is activated.

The source of the root key that IMS supplies to the randomizing routine is as follows:

- For a root insert, it is taken from the I/O area containing the root to be inserted.
- For a call qualified on the root key, it is the key value in the segment search argument.

Related Reading: For information on processing Get Next (GN) calls qualified on the root key and calls with root qualification that allow a range of key values, see *IMS/ESA Application Programming: Database Manager*.

The key is supplied to the randomizing module for conversion to a relative block number and anchor point number within the database. In addition to the key supplied by an application program, parameters from the DBD generation associated with the database being used are available to the randomizing module.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the routine.

Contents of Registers on Entry

Upon entry to the randomizing module, the registers contain the following:

Register	Contents
0	Number of entries in the MRMB (total number of AREAs in the DEDB).
1	Address of first MRMB the routine uses.
2	Size of an entry in the MRMB.
3	Address of the root key.
4	Length of the root key in bytes.
5	Total number of RAPs in the DEDB.
6	Address of an eight-word area that the randomizing module can use.
10	Address of the EPST (Extended Partition Specification Table).
11	Address of the ESCD (Extended System Content Directory).
13	Address of save area. The routine must not change the first three words
14	Return address to IMS.
15	Entry point of randomizing module.

The randomizing module must neither change the key value nor modify any control blocks.

Exception: When you execute MVS batch utilities (such as DBFUCDX0 or MSDB-to-DEDB conversion), register 10 contains zeros and register 11 contains decimal -1 (X'FFFFFFFF').

Description of Parameters

To support the facility of randomizing within an AREA, the routine is passed the address of a Randomizing Module Block (MRMB).

Each AREA has one 3-word entry. MRMB entries are built in the same order as their associated AREA macros in the DBDGEN for the database. The content of an entry is mapped by DBFMRMB macro and contains the following:

```
MRMB      DSECT
MRMBARTD      DS      0F   START IS WORD-ALIGNED
MRMBARTC      DS      F    ADDRESS OF THE AREA SELECTED
MRMBARTI      DS      F    NUMBER OF ANCHOR POINTS IN THIS AREA
MRMBARTN      DS      F    CUMULATIVE NUMBER OF ANCHOR POINTS
*            IN ALL AREAS OF THE DEDB UP TO AND
*            INCLUDING THIS ONE
MRMBARTZ      DS      0F   END OF THIS ENTRY, START OF NEXT
MRMBARTL      EQU     MBMBARTZ-MRMBARTD
*            LENGTH OF A SINGLE ENTRY
```

Contents of Registers on Exit

Before returning to IMS, your routine must restore all registers, except for registers 0, 1, and 15, which must contain the following:

Register	Contents
0	Relative root anchor point number within the selected AREA (0 for first root anchor point).
1	DMAC address of the AREA selected.
15	Return code interpreted as follows:

Return Code	Meaning
0	Register 1 contains the address of the area selected. If the area is not contained in the DMCB or the HSSP sublist, ABENDU1021 is issued.
4	Status 'FM' needs to be issued.

Any other return code causes ABENDU1021 to be issued.

When randomizing through the entire DEDB, the randomizing module must derive an AREA and a relative root anchor point number to conform to the exit interface described above. You can use the third word of the MRMB entry to accomplish this.

Using the Sample DEDB Randomizing Routines (DBFHDC40)

The sample exit routine is based on the generalized Randomizing Routine (DFSHDC40) and has been modified to work with DEDB databases. The modifications are:

1. The module uses the DEDB input and output interfaces.
2. The module can return an anchor point in block 1, because DEDB areas do not use a bit map at this location.

Extended Call Interface (XCI) Option

There is now an extended call interface (XCI) option that can be specified in the RMNAME= parameter list in the DBD statement of a DBDGEN

The XCI option specifies that this DEDB uses the extended call interface when making calls to the randomizer. This option allows the randomizer to be called in 3 different ways. On initialization of IMS, or during a `/START DB` command, IMS will first load the randomizer and then make an 'INIT' call to the randomizer to invoke its initialization routines. During a `/DBR DB` command, IMS will make a 'TERM' call to the randomizer to invoke the termination routines before unloading the randomizer. The normal randomizing call is made when the application issues a GU or ISRT call on a root segment. The XCI option is valid only for DEDBs.

Attributes of the Routine

The attributes of the routine are the same as the non-XCI randomizer.

Invoking the Routine

An XCI randomizer is invoked with an initialization call during Fast Path initialization and during a `/START DB` command. The XCI randomizer is invoked with a termination call during a `/DBR DB` command. Otherwise, a regular randomizing call is made to the XCI randomizer when an application program issues a GU or ISRT call which operates on a root segment of a DEDB database, just as in a non-XCI randomizer.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the routine.

Contents of Registers on Entry for a Randomizing Call

Register	Contents
0	'XCI '
1	address of parameter list:

```
   4    4     4     4     4     4     4     4     4 bytes
------------------------------------------------------
|  0 | (a) | (b) | (c) | (d) | (e) | (f) | (g) | (h) |
------------------------------------------------------
(a) - number of areas
(b) - address of MRMB
(c) - size of MRMB
(d) - address of key
(e) - key length
(f) - total RAPs
(g) - address of work area
(h) - userdata
```

Contents of Registers on Entry for an Initialization Call

Register	Contents
0	'XCI '
1	address of parameter list:

```
   4    4     4     4 bytes
------------------------
|  4 | (a) | (b) | (c) |
------------------------
(a) - address of DMCB
(b) - address of ECB
(c) - userdata
```

Contents of Registers on Entry for a Termination Call

Register	Contents
0	'XCI '
1	address of parameter list:

```
   4    4     4     4 bytes
------------------------
|  8 | (a) | (b) | (c) |
------------------------
(a) - address of DMCB
(b) - address of ECB
(c) - userdata
```

Contents of Registers on Exit from a Randomizing Call

The contents of registers on exit from a randomizing call are the same as the contents for a non-XCI randomizer.

Contents of Registers on Exit from an Initialization Call

Register	Contents
1	Reason code for a non-zero return code.

Register	Contents
15	Return code.

Contents of Registers on Exit from a Termination Call

Register	Contents
1	Reason code for a non-zero return code.
15	Return code.

Chapter 5. Data Entry Database Resource Name Hash Routine (DBFLHSH0)

This chapter describes the IMS Data Entry Database (DEDB) Resource Name hash routine. The chapter discusses the attributes of the routine and how the routine communicates with IMS.

In this Chapter:

About This Routine

The IMS DEDB Resource Name hash routine is used with the Internal Resource Lock Manager (IRLM) and enables IMS and DBCTL to maintain and retrieve information about the control intervals (CIs) used by sharing subsystems. The routine does this by performing a hashing function on the high-order three bytes of the relative byte address (RBA) representing a CI and using the hashing result as a displacement into the hash table. If you are using IRLM in your system, the routine IMS supplies (DBFLHSH0) or the replacement routine that you write yourself is called automatically.

Related Reading: For guidance-level information to help you decide whether to write your own hashing routine, see the section on "Factors Influencing Fast Path Performance" in *IMS/ESA Administration Guide: Database Manager*.

This routine must be written and link-edited as *reentrant* (RENT) like the one supplied by IMS. It receives control and must return control in 31-bit addressing mode. It must be able to execute in cross-memory and TASK modes.

Attention: All IMS systems sharing data must use the same routine or DEDBs could be destroyed. IMS does not check to ensure that the routines are the same.

Also, the routine supplied with version 5.1 of IMS does not match the routine supplied with IMS versions 3.1 and 4.1. If you want to share any DEDBs between an IMS version 5.1 system and a system using IMS version 3.1 or 4.1, ensure that all systems are using the same hashing routine.

Attributes of the Routine

Table 12 shows the attributes of the Data Entry Database Resource Name Hash routine.

Table 12. Data Entry Database Resource Name Hash Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL
Naming convention	You must name this exit routine DBFLHSH0.
Link editing	Follow the guidelines described in "Link-Editing the Routines" on page 5. After you compile and test the routine, link-edit it into IMS.RESLIB or to the library specified in the USERLIB= parameter of the IMSGEN macro statement.

Table 12. Data Entry Database Resource Name Hash Routine Attributes (continued)

Attribute	Description
Including the routine	At system definition time, you must specify the name of your routine in the USRHASH parameter of the FPCTRL macro. **Related Reading:**For details, see the section on the FPCTRL macro in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.
IMS callable services	This exit is not eligible to use IMS Callable Services.
Sample routine location	IMS.DBSOURCE (member name DBFLHSH0)

Assembling the Routine

In a multiple-IMS environment, all IMS systems must use the same hashing routine and compile that routine at the same time. If you write your own routine, you must store the compile time in the module using &SYSDATE and &SYSTIME. You also must place the address of the date and time in the first field of the routine's CSECT.

Communicating with IMS

IMS uses the entry registers and parameter list, and the exit registers to communicate with the routine.

Contents of Registers on Entry

Upon entry, the routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of Extended Partition Specification Table (EPST).
13	Address of save area. The routine must not change the first three words.
14	Return address to IMS.
15	Entry point of hash routine.

Description of Parameters

As input to the hashing routine, you need to supply *one* of the following:

- the high-order byte of an RBA.
- the names of both a database and an area.

The routine performs an EXCLUSIVELY OR on this input, stores it in a field, and returns a hash value result to the field EPSTRSHS.

EPST Input to the Routine

Register 1 points to the extended program specification table (EPST) that contains this input as follows:

Field Name	Content
EPSTRSHS	Hashing routine result. Only the low-order 14 bits are significant.
EPSTRSID	Start of the lock name to be hashed. Lock resource name consists of:

Name	Meaning
EPSTLKID	A lock identifier. If EPSTLKID = 0, resource name is for CI. If EPSTLKID = nonzero, name is for the area. 1 byte.
EPSTRBA	Bit 0 through 23 of RBA. 3 bytes.
EPSTDMCB	DB Number as defined by DBRC. 2 bytes.
EPSTAREA	Area number. 1 byte.
EPSTDBNM	Database name. 8 bytes.
EPSTARNM	Area name. 8 bytes.

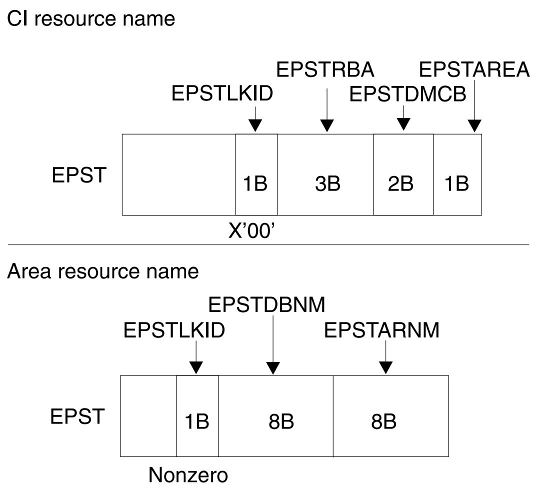


Figure 3. Lock Resource Name

EPST DSECT

The DSECT of the extended program specification table (EPST) (name: DBFEPST), and the DEDB area control list (DMAC) (name: DBFDMAC) can be used. The DMAC address is set at the EPSTDMAA field.

Sample Hashing Routine Result Format

The following shows the layout of the hash value stored in EPSTRSHS using the IMS-supplied routine DBFLHSH0:

```
A    -  Bits 0 - 17 of EPSTRSHS   -  18 bits
B    -  Bits 21 - 25 of CI RBN
          XOR'd COMB value    -    5 bits
C    -  Bits 26 - 29 of CI RBN ¹    -    4 bits
D    -  Bits 16 - 20 of CI RBN
         XOR'd COMB value ²    -    5 bits
```

Notes:

1. COMB VALUE (bits 3 - 7) = bits 11 - 15 of DMCB XOR'd with bits 7, 6, 5, 4, and 3 of the area number.
2. CI RBN = RBA divided by the CI size.

Chapter 6. Data Entry Database Sequential Dependent Scan Utility Exit Routine (DBFUMSE1)

This chapter describes the Data Entry Data Base (DEDB) Sequential Dependent Scan Utility exit routine. The chapter provides information about the attributes of the routine, how the routine is called, and how the routine communicates with IMS. A sample routine is provided at the end of the chapter.

In this Chapter:

About This Routine

The DEDB Sequential Dependent Scan utility allows you to write an exit routine that copies and processes a subset of the number of segments that the utility scans at a particular time. For example, such a routine might change both the content and length of the segments scanned. You can choose to sort or not to sort the segments.

If you do not write an exit routine, the Scan utility defaults to passing unchanged segment contents through the range you have specified for scanning. If you do not specify a limit on the range of segments that the utility can scan, the utility scans and copies all of the dependent segments.

In-doubt segments are not passed to this exit routine.

Related Reading: For guidance-level information to help you determine whether to write an exit routine for use with the Scan utility, see *IMS/ESA Utilities Reference: Database Manager*.

This routine must be written and link-edited as *reentrant* (RENT) like the one supplied by IMS. The exit routine receives control and must return control in 24-bit addressing mode. It must be able to execute in cross-memory and TASK modes.

Attributes of the Routine

Table 13 shows the attributes of the Data Entry Database Sequential Dependent Scan Utility exit routine.

Table 13. Data Entry Database Sequential Dependent Scan Utility Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	This exit routine has no specific naming requirements or restrictions; standard naming conventions apply. **Related Reading:**For information about general guidelines, see "Naming the Routines" on page 4 in "Chapter 1. Guidelines for Writing IMS Exit Routines" on page 3.

Table 13. Data Entry Database Sequential Dependent Scan Utility Exit Routine Attributes (continued)

Attribute	Description
Link editing	Follow the guidelines described in “Link-Editing the Routines” on page 5. After you compile your routine, include it into IMS.RESLIB or into any operating system partitioned data set to which access is provided with a JOBLIB or STEPLIB control region JCL statement.
Including the routine	No special steps are needed to include this routine.
IMS callable services	This exit routine is not eligible to use IMS Callable Services.
Sample routine location	Refer to “Sample DEDB Sequential Dependent Scan Utility Exit Routine (DBFUMSE1)”.

Calling the Routine

If you want IMS to call your routine instead of the IMS-supplied routine (DBFUMSE0), you must specify the name of your routine in the EXIT control statement of the SYSIN DD data set of the Scan Utility JCL.

Related Reading: For details, see *IMS/ESA Utilities Reference: Database Manager*.

Communicating with IMS

IMS uses the entry registers, parameter list, and exit registers to communicate with the routine.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of parameter list.
13	Address of save area. The exit routine must not change the first three words.
14	Return address of IMS.
15	Entry point of exit routine.

Contents of Registers on Exit

Before returning to IMS, the routine must restore all registers except for register 15, which must contain one of the following:

Return Code	Meaning
00	Use segment.
04	Do not use segment.

Sample DEDB Sequential Dependent Scan Utility Exit Routine (DBFUMSE1)

The following sample is *not* a usable exit routine provided by IMS nor is it found in IMS.DBSOURCE library. It is an example showing entry and exit code to help you write your own scan routine.

```
   TITLE 'DBFUMSE1 IMS DEDB ONLINE UTILITY SCAN EXIT'
***********************************************************************
*                                                                     *
*  MODULE NAME : DBFUMSE1                                             *
*                                                                     *
*  TITLE : STANDARD EXIT FROM SCAN UTILITY                            *
*                                                                     *
* CONTAINS RESTRICTED MATERIALS OF IBM                                *
*  COPYRIGHT : REFERENCE MODULE DBFCOPYR                              *
*                                                                     *
*  ENTRY POINT(S)/PURPOSE : DBFUMSE1                                  *
*                                                                     *
*  FUNCTION : THIS IS A SAMPLE OF THE SCAN UTILITY USER EXIT.         *
*             ITS PURPOSE IS TO DEFINE THE INTERFACE BETWEEN          *
*             THE UTILITY AND THE EXIT.  IT IS NOT INTENDED TO        *
*             BE A USABLE EXIT.  IN THIS EXAMPLE, OUTPUT TO THE       *
*             SCAN DATASET IS SUPPRESSED IF THE SEGMENT BEGINS        *
*             WITH HEX ZEROES.                                        *
*                                                                     *
*  ENTRY INTERFACES:                                                  *
*                                                                     *
*   REGISTERS AT ENTRY : R1 ADDRESS OF USER PARAMETER LIST            *
*                        R13 ADDRESS OF SAVE AREA                     *
*                        R14 ADDRESS OF RETURN POINT                  *
*                        R15 ADDRESS OF ENTRY POINT                   *
*                        REGISTERS ARE SAVED AND RESTORED BY THE      *
*                            CALLING MODULES.                         *
*                                                                     *
*   CONTENT OF PARAMETER LIST (UTDWUSER) :                            *
*             UTDWDATA - ADDRESS OF SEGMENT (FULLWORD)                *
*                                 ZERO AFTER LAST SEGMENT             *
*                                 1. AT ENTRY ADDRESS OF SEGMENT      *
*                                 2. AT EXIT ADDRESS OF DATA TO BE    *
*                                    PICKED UP AND PUT INTO SCAN      *
*                                    OUTPUT DATA SET REFERRED TO      *
*                                    BY SCANCOPY DD CARD.             *
*             UTDWMIN -  MINIMUM LENGTH OF SEGMENT (HALFWORD)         *
*                                 AS IN DBD-GENERATION                *
*             UTDWMAX -  MAXIMUM LENGTH OF SEGMENT (HALFWORD)         *
*                                 AS IN DBD-GENERATION                *
*             UTDWUFLD - FIELD FOR USER (FULLWORD)                    *
*                                 ZERO WITH FIRST SEGMENT,            *
*                                 UNCHANGED BY THE UTILITY            *
*             UTDWMOUT - MAXIMUM SEGMENT LENGTH (HALFWORD)            *
*                                                                     *
*                        NOTE: THE USER MAY CHANGE LENGTH AND         *
*                                 CONTENT OF THE SEGMENT USING HIS    *
*                                 OWN WORKSPACE. IF HOWEVER THE LENGTH*
*                                 EXCEEDS THE LENGTH OF THE SCAN      *
*                                 OUTPUT BUFFER - 8 THE UTILITY IS    *
*                                 TERMINATED.                         *
*                                                                     *
*   DATA/OTHER : NONE                                                 *
*                                                                     *
*  EXIT INTERFACES :                                                  *
*   REGISTERS AT EXIT : R15 CONTAINS RETURN CODE                      *
*   RETURN CODES :   00  USE SEGMENT                                  *
*                    04  DO NOT USE SEGMENT                           *
*                                                                     *
*    DATA/OTHER :  NONE                                               *
*                                                                     *
*  EXTERNAL ROUTINES CALLED : NONE                                    *
*                                                                     *
*  TABLES/WORKAREAS : NONE                                            *
*                                                                     *
*  REGISTER USAGE : R1  PARAMETER LIST                                *
*                   R2  SEGMENT ADDRESS                               *
*                   R12 MODULE BASE REGISTER                         *
*                   R14 RETURN ADDRESS                               *
*                   R15 RETURN CODE - 00  WRITE SEGMENT              *
*                                     04  DO NOT WRITE SEGMENT       *
*                                                                    *
*  MESSAGE NUMBERS :  NONE                                           *
*                                                                    *
*  ABEND CODES : NONE                                                *
*                                                                    *
**********************************************************************
         EJECT ,
*PCODE:
**********************************************************************
*                                                                    *
*     IF SEGMENT EXISTS                                              *
*        IF THE SEGMENT STARTS WITH X'00'S                           *
*           SET RC=4  (DON'T WRITE THE SEGMENT)                      *
*        ELSE                                                        *
*           SET RC=0  (WRITE THE SEGMENT)                            *
*        ENDIF                                                       *
*     ELSE                                                           *
*        SET RC=4  (DON'T WRITE THE SEGMENT)                         *
*     ENDIF                                                          *
*     RETURN                                                         *
*                                                                    *
**********************************************************************
*ENDPCODE:
         SPACE 10
         PRINT NOGEN
         REQUATE
         DBFUTDW                   DSECT FOR PARM LIST
         SPACE 10
DBFUMSE1 CSECT
         USING DBFUMSE1,R12        MODULE BASE REGISTER
         USING UTDWUSER,R1         PARAMETER LIST BASE REGISTER
         L     R2,UTDWDATA         GET ADDRESS OF SEGMENT
         LTR   R2,R2               IS THERE A SEGMENT?
         BZ    NOWRITE             NO SEGMENT, DON'T WRITE
*
         LA    R2,2(,R2)           SKIP PAST SEGMENT LENGTH
         CLC   0(6,R2),ZEROES      DOES SEGMENT START WITH 0'S?
         BNE   WRITESEG            NON-ZERO DATA.  WRITE IT.
*                                  OTHERWISE, DON'T WRITE IT.
NOWRITE  DS    0H
         LA    R15,4
         BR    R14
WRITESEG DS    0H
         XR    R15,R15
         BR    R14
ZEROES   DC    XL6'00'
         END
```

Chapter 7. HDAM Randomizing Routines (DFSHDC40)

This chapter describes the HDAM randomizing routines. The chapter provides information about the attributes of the routines, how the routines are called, and how the routines communicate with IMS.

For the latest version of the sample routine (DFSHDC40), see the IMS.SOURCE library; member name is DFSHDC40.

In this Chapter:

About These Routines

The DL/I HDAM access method requires you to supply a randomizing module for placing root segments in, or retrieving them from, an HDAM database. Several databases can share the same routine, but each of those databases must be associated with a single randomizing routine. If you are using data sharing, you must use the same randomizing routine for both systems.

A randomizing module uses a mathematical technique to convert a key into an address. The same key always converts to the same address. The randomizing module required by IMS must convert a key field value into a relative block number and an anchor point number. The result of a randomizing routine is a relative block number that ranges from 1 to $2^{24}-1$. The anchor point number ranges from 1 to 255.

The key field value is supplied by an application program in the data itself for inserting segments into the database and in an application program in an SSA (segment search argument) for retrieving segments from a database.

Four randomizing modules are supplied with IMS. Although four are supplied, DFSHDC40 is the only one recommended for use. You can use this one or write your own randomizing module.

Related Reading: To help you determine the module that best meets your need, see *IMS/ESA Administration Guide: Database Manager*.

If you write your own module, follow the guidelines included in this chapter.

Attributes of the Routine

You must write, compile, and link-edit the randomizing module as one of the following:

REENTRANT	IMS does not serialize the database before calling the routine. A single copy of the routine is used for the databases.
REUSABLE	IMS serializes the database before calling the routine. If the routine is used for multiple databases, it must be written and compiled as reentrant, even if it is not link-edited as reentrant.

NONREUSE	IMS serializes the database before calling the routine. Each database has its own copy of the routine.

All modules receive control and must return control in 31-bit addressing mode. They must be able to execute in cross-memory and TASK modes.

Naming the Routine

The name you give to the load module used for randomizing functions with a specific database must appear in the DBD generation associated with the database. The load module name must be the value of the “mod” parameter of the RMNAME= operand on the DBD statement in the HDAM DBD generation.

Related Reading: For details on coding this parameter, see Chapter 1, “DBD Generation”, in *IMS/ESA Utilities Reference: Database Manager*.

Link-Editing the Routine

After you compile and test a randomizing module, link-edit it into IMS.RESLIB, SYS1.LINKLIB, or into any operating system partitioned data set that can be accessed by a JOBLIB or STEPLIB JCL statement for the IMS control, SAS, and batch regions.

To ensure that the routines run as they did in prior IMS releases, link-edit them as neither reentrant nor reusable.

Loading the Routine

IMS loads all randomizing modules from their resident library when the database is opened. IMS obtains the name of the randomizing module from the name you have specified in the RMNAME parameter of the DBD statement of the database description (DBD).

Related Reading: For details on coding the RMNAME parameter, see *IMS/ESA Utilities Reference: Database Manager*.

The necessary randomizing module associated with a specific database is brought into main storage at the time the associated database is opened. It can also be placed in the LPA (linkpack area). This allows one copy of the module to service several databases that are concurrently open.

If you use any of the Local Storage Options (LSO), the randomizing module is loaded in CTL or DL/I SAS private storage. Otherwise, the module is loaded into CSA.

Related Reading: For more information about general guidelines, see “Chapter 1. Guidelines for Writing IMS Exit Routines” on page 3.

Calling the Routine

When an application program issues a Get Unique or Insert call that operates on a root segment of an HDAM database, the randomizing module is called.

The source of the root key that IMS supplies to the randomizing routine is as follows:

- For a root insert, IMS takes the key from the I/O area containing the root to be inserted.

- For a call qualified on the root key, IMS uses the key value in the segment search argument.

Related Reading: For information on processing Get Next (GN) calls qualified on the root key and calls with root qualification that allows a range of key values, see *IMS/ESA Application Programming: Database Manager*.

The key is supplied to the randomizing module for conversion to a relative block number and anchor point number within the database. In addition to the key supplied by an application program, parameters from the DBD generation for the database are available to the randomizing module.

Using IMS Callable Services with This Routine

This exit routine is not eligible to use IMS Callable Services.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the randomizing routine.

Contents of Registers on Entry

Upon entry, the randomizing routine must save all registers using the provided save area. The registers contain the following:

Register	Content
0	Address of Data Management Block (DMB).
1	Address of the DMBDACS CSECT.
7	Address of Partition Specification Table (PST).
9	Address of first byte of key field value supplied by an application program.
13	Address of save area. The exit routine must not change the first three words.
14	Return to IMS address.
15	Entry point of randomizing module.

If an HDAM database does not have a sequence field defined:

- The executable key length field in the CSECT named RDMVTAB is not initialized and must not be used.
- The value in register 9 module contains the address of the first byte of the segment.

If an HDAM database does not have a sequence field defined at the root level, the randomizing module is given control on an insert call. All retrieval calls result in a scan of the root-level qualification. On Get Unique (GU) calls, the scan starts at the beginning of the database. On Get Next (GN) calls, the scan starts at the current root-level position within the database.

The randomizing module is invoked on Get calls, particularly when the database contains a secondary index or a logical relationship. The randomizing module must produce the same results on the Get call as it did on the Insert call.

The first eight words of the PST are available to the randomizing module as a work area. These words are also used by DL/I and must not be used by other exit routines. If an additional work area is needed, CSECT RDMVTAB can be expanded to provide additional space.

HDAM Randomizing Routines

Internal IMS control blocks that can be of value to a randomizing routine are the Partition Specification Table (PST), the Physical Segment Description Block (PSDB) for the root segment, and the first Field Description Block (FDB). The FDB is the root segment key field format description. DSECTs of these three blocks are provided in the examples shown later in this chapter.

Description of Parameters

The parameters from DBD generation are available to randomizing modules. Their area is described by the DMBDACS DSECT. It contains information such as the randomizing routine's name, anchor point information, and the total area length. You can extend the area by an assembly and link-edit process to contain any data or algorithm information.

The root 32 bytes of the RDMVTAB CSECT (described by the DMBDACS DSECT) contains constants defined by DBDGEN. If you extend the area to include additional parameters, this field must be duplicated. The DMBDASZE field must be updated to reflect the total length of this area (including the added parameters).

After assembly, the expanded RDMVTAB CSECT can be link-edited to replace the old one. Use an ENTRY statement specifying the name of the DBD and an ORDER statement to make sure the original order of the multiple CSECTs is maintained. For more information, see information on the MVS linkage editor and loader in the MVS/ESA product library.

The following DSECT defines the format of the area pointed to by register 1:

```
DMBDACS   DSECT
DMBDANME  DS     CL8   NAME OF ADDR ALGORITHM LOAD MODULE
DMBDAKL   DS     CL1   EXECUTABLE KEY LENGTH OF ROOT
          DS     CL3
DMBDASZE  DS     H     SIZE OF THIS CSECT
DMBDARAP  DS     H     NUMBER OF ROOT ANCHOR POINTS/BLOCK
DMBDABLK  DS     F     NUM OF HIGHEST BLOCK DIRECTLY ADDRSD
DMBDABYM  DS     F     MAX NUMBER OF BYTES BEFORE OFLOW TO
                       2NDARY
DMBDARC   DS     CL1   RETURN CODE FROM RANDOMIZER
          DS     CL3   RESERVED
DMBDACP   DS     F     RESULT OF LAST ADDRESS CONVERSION
```

Contents of Registers on Exit

Before returning to IMS, the randomizing routine must restore all registers. The parameter list pointed to by register 1 and shown in Description of Parameters can contain one of the following return codes:

Return Code	Meaning
0	Continue processing; randomizing properly.
4	Set FM status code and return to caller.
8	U812 abend.

For any randomizing routine that passes these return codes, ensure that application programs that use the database can accept the return codes.

The return code from a randomizing module can be in either character or binary form. In other words, X'F0' and X'0' are both valid for a return code of zero. This return code must be placed in the DMBDARC field of the CSECT addressed by register 1.

You do not need to explicitly set a return code of zero in DMBDARC, because it is the default return code and the field is preset to zero.

Results of the Routine on Exit

The result of a randomizing module conversion must be in the form `BBBR` where:

BBB	Is a 3-byte binary number of the block into which a root segment is inserted or from which it is retrieved.
R	Is a 1-byte binary number of the appropriate anchor point, within a relative block, within a data set of the database.

This result must be placed in the CSECT addressed by register 1 in the 4-byte fixed name DMBDACP. If the result exceeds the content of the field DMBDABLK, the result is changed to the highest block and last anchor point of that block.

Sample HDAM Randomizing Routines

IMS supplies four randomizing module samples (DFSHDC10, DFSHDC20, DFSHDC30, and DFSHDC40) to help you write your own HDAM randomizing module. The modules are linked into the IMS.RESLIB data set during system definition. The modules use the following randomizing techniques:

- Modulo or division method (DFSHDC10)
- Binary halving method (DFSHDC20)
- Hashing method (DFSHDC30 and DFSHDC40)

Module DFSHDC40 is recommended; the source code for all four modules resides in the IMS.DBSOURCE library. The next section provides guidelines for using the sample module, DFSHDC40.

Restriction: These routines do not support nonsequenced HDAM databases. They all use the key length in their calculations.

Using the Sample HDAM Generalized Randomizing Routine (DFSHDC40)

If root keys are unique and totally random storage is desired, this routine can be used for **any** HDAM database without performing an analysis of key distributions.

This randomizing routine works with the entire key and has the following characteristics:

- It is reentrant.
- Keys can contain any of the 256 characters, and key length can be from 1 to 256 bytes.
- It converts **any** key distribution (with unique key values) to a totally random address distribution.
- It never returns an address in block 1, which is always a bit map block in HDAM. The user can specify any number of blocks and RAPs.
- The number of blocks must be in the range between 2 and (2**24)−1; the number of RAPs must be in the range of 2 to (2**31)−1 when RAPs are multiplied by blocks. The RBN subparameter of the RMNAME= parameter of the DBD statement must be specified for the upper limit, together with DFSHDC40 as the “mod” subparameter, if this randomizing routine is chosen.

- It allows the insertion of a dummy root at the highest block-RAP to ensure the formatting of the entire root addressable area at load time.

The basic logic of the routine is:

1. Convert the key into a 4-byte binary number by translating the key digits twice. Determine the offset into the translation table using the key length and individual digits. For example:

   ```
   Key      123456
   ```

   Digits are used in series of threes. Two work areas are used. In the first pass, the first work area contains X'F2F3'; the second contains X'F1F2F3'.

   The first work area is translated into the translation table with a zero point of 4 (key length 2). The second work area is translated into the translation table with a zero point of X'F5', the fifth digit. These two translated numbers are multiplied and added into an accumulator. The remaining digits are converted and added into the accumulator.

   The conversion number for key 123456 is X'45683199'.
2. Translate the converted number, and set the top bit to zero to ensure a positive number.
3. Multiply the maximum number of blocks minus one by the number of RAPs. Multiply the result by the translated key.
4. After adjustment to ensure block 1 is not used, store the result in DMBDACP.

For the latest version of DFSHDC40 see the IMS.DBSOURCE library; member name is DFSHDC40).

Chapter 8. Secondary Index Database Maintenance Exit Routine

This chapter describes the Secondary Index Database Maintenance exit routine. It provides information about the attributes of the routine, how the routine is called, and how the routine communicates with IMS.

In this Chapter:

About This Routine

Two options are available to the Database Manager to control the volume of entries in secondary index databases: the NULLVAL operand and the index maintenance exit routine. To build and maintain a sparse index, you can use suppression of indexing, the process of withholding a prospective index pointer segment from the index.

Use the NULLVAL operand to suppress indexing when the entire indexed field contains one specified character or value. For example, you may want to use NULLVAL to suppress indexing when the indexed field contains only blanks. A different NULLVAL can be specified for each indexed segment.

Alternatively, secondary indexing allows you to specify, during the DBDGEN, a user-supplied exit routine that can selectively suppress secondary indexing. You can thereby control the density of a secondary index. One exit routine is allowed for every secondary index; however, one generalized routine can be written to serve several index relationships.

If this exit routine is link-edited as reentrant (RENT), it must be truly reentrant (it cannot depend on **any** information from a previous invocation and it cannot store into itself).

If this exit routine is link-edited as reusable (REUSE), it must be truly reusable (it cannot depend on **any** information in itself from a previous call), but it can depend on information that it saves in the specific database segment block that is passed to it. In addition, if the same exit routine is used for two different segments, the single copy of the exit can be called concurrently for each segment. In this case, the exit routine must be written as reentrant.

If this exit routine is link-edited so that it is neither RENT nor REUSE, it can store into itself and depend on the information saved in the database segment block that is passed to it.

Table 14 shows the attributes of the Secondary Index Database Maintenance exit routine.

Table 14. Secondary Index Database Maintenance Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.

Table 14. Secondary Index Database Maintenance Exit Routine Attributes (continued)

Attribute	Description
Naming convention	Each exit routine must have a name unique with respect to all IMS module names and to any other exit routines in the IMS libraries. The name corresponds to the name specified in the EXTRTN subparameter, in the XDFLD statement, for the DBD generation. **Related Reading:**For details about this subparameter, see the section on DBD generation in *IMS/ESA Utilities Reference: System*. Before an index source segment in a database can be loaded or updated, its EXTRTN routine must be in the system library.
Link editing	Follow the guidelines described in “Link-Editing the Routines” on page 5. After an exit routine has been compiled and tested, it can be placed into the IMS.RESLIB data set, from which it is loaded by IMS. It can also be placed in SYS1.LINKLIB, or in any operating system partitioned data set to which access is provided with a JOBLIB or STEPLIB JCL statement.
Including the routine	No special steps are need to include this routine.
IMS callable services	This exit routine is not eligible to use IMS Callable Services.
Sample routine location	Refer to “Sample Secondary Index Database Maintenance Exit Routine” on page 90.

Loading the Routine

The exit routine associated with the specific database is loaded into storage in either the IMS online control program region or batch processing region when the associated database is opened. This allows one copy of the module to service several databases that are open concurrently.

When an index maintenance exit routine is used in either the IMS online control region or a DL/I batch processing region and the exit routine does not exist in LINKPACK, you must provide space in the IMS control region or in the DL/I separate address space (DLISAS) to accommodate the exit routines that can be used for online databases.

Calling the Routine

When an application program issues a REPL, ISRT, or DLET call of a segment serving as an index source segment for one or more indexing relationships, the DL/I index maintenance routine is invoked.

DLET Call

In the case of DLET, an indexing segment is built corresponding to the existing index source segment. If it passes the null value test, the index exit routine is invoked. This routine indicates whether this indexing segment should appear in the index. If it should appear, the actual indexing segment is retrieved and deleted; otherwise, no delete is attempted.

ISRT Call

In the case of ISRT, the indexing segment is built to correspond to the segment to be inserted, and the null value test and the exit routine tests are performed. If no suppression of indexing is indicated by either, it is inserted into the index.

REPL Call

A REPL call can be a combination of the above, a simple replace, or a NOP, depending on the fields changed in the replace. If a field in the Index Source

Segment (ISS) is changed by a REPL call that changes the indexed data or subsequent data, the existing indexing segment is deleted and a new one inserted. The index edit routine is invoked for each operation. If the change in the ISS affects a source data field, a replace operation on the indexing segment is executed, unless the index exit routine indicated that indexing was suppressed. If the ISS replace made no changes in the indexing segment, no action is taken.

The suppression of indexing by the exit routine must be consistent. The same indexing segment cannot be examined at two different times and have suppression indicated only once. If the indexing segment contains user data, this user data cannot be used to evaluate suppression, since the actual indexing segment is seen by the exit routine just before the insertion of a new one. In the cases of replace and delete, only a prototype is passed. The prototype contains the constant, indexed data, subsequence data, duplicate data, and any symbolic pointer that was added. Therefore, index suppression must not be based on any user data.

Parameters to be passed to the index routine are indicated later in “Description of Parameters”. By issuing a return code, the exit routine indicates either that the present index pointer segment belongs in the index or that it should be suppressed. The exit routine must not change any IMS control blocks, or any fields in the indexing segment.

You can include additional information about the segment in the exit routine CSECT. This CSECT is part of the DBD, and as such can be replaced by a link-edit. It is of variable-length and contains a fixed-format header. A separate CSECT is provided for each XDFLD in the DBD for which an exit routine is specified. The availability of this CSECT is described in the exit routine specifications. You can replace this control section in the same manner as you can the segment compression control section.

For additional information, see the section Description of Parameters on page 100 that is included with the description of the Segment Edit/Compression routine.

Communicating with IMS

IMS communicates with the exit routine through the entry and exit registers.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of Partition Specification Table (PST).
2	Address of proposed or existing index segment.
3	Address of Index Maintenance Routine Parameters CSECT.
4	Address of Index Source Segment.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of exit routine.

Description of Parameters

Upon entry to the routine, IMS passes the address of the following CSECT:

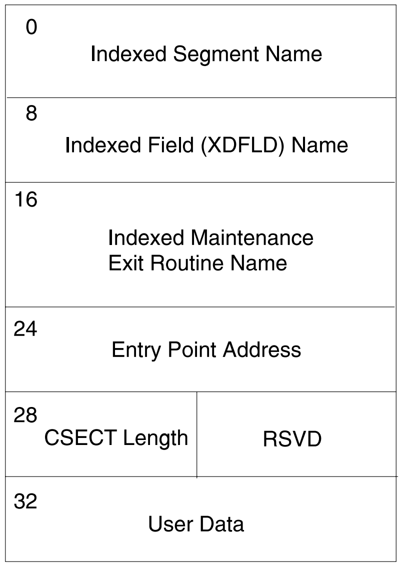


Figure 4. Index Maintenance Exit Routine Parameter List CSECT

The following DSECT defines the format of this CSECT:

```
DMBXMPRM    DSECT
DMBXMSGN    DS      CL8     Name of indexed segment
DMBXMXDN    DS      CL8     Name of indexed field
DMBXMXNM    DS      CL8     Name of exit routine
DMBXMXEP    DS      A       Entry point addr
DMBXMPLN    DS      H       Total length of CSECT
            DS      H       Not Used
```

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers except register 15, which contains one of the following return codes:

Return Code	Meaning
0	The indexing segment should appear in the index for this database segment.
4	Indexing should be suppressed.

Sample Secondary Index Database Maintenance Exit Routine

The following sample is **not** a usable exit routine provided by IMS, nor is it found in the IMS.DBSOURCE library. It is an example showing entry and exit code to help you write your own routine.

```
SAMPLE   TITLE 'SAMPLE OF SECONDARY INDEX EXIT ROUTINE'
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                     *
* SAMPLE OF SECONDARY INDEX DATA BASE MAINTENANCE EXIT ROUTINE        *
*                                                                     *
* THIS SAMPLE IS NOT INTENDED TO BE A USABLE EXIT ROUTINE.            *
* IT IS PROVIDED HERE TO SHOW ENTRY AND EXIT CODE.                    *
* THIS SAMPLE SUPPRESSES THE INDEX ENTRY IF ALL BYTES OF THE          *
* INDEX KEY ARE BLANK.                                                *
*                                                                     *
*                                                                     *
*    REGISTERS ON ENTRY                                               *
*      R1  - PARTITION SPECIFICATION TABLE (PST) ADDRESS              *
*      R2  - ADDRESS OF (PROPOSED OR EXISTING) INDEX SEGMENT          *
*      R3  - ADDRESS OF INDEX MAINTENANCE ROUTINE PARMS CSECT         *
*      R4  - ADDRESS OF INDEX SOURCE SEGMENT                        *
*      R13 - SAVE AREA ADDRESS                                      *
*      R14 - RETURN ADDRESS                                         *
*      R15 - ENTRY ADDRESS                                          *
*                                                                   *
*    REGISTERS ON EXIT                                              *
*      R15 - 0 TO NOT SUPPRESS THE INDEX ENTRY                      *
*          - 4 TO SUPPRESS THE INDEX ENTRY                          *
*      R0 THRU R13 ARE RESTORED                                     *
*                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         SPACE 1
INDEXXIT CSECT
         STM   R14,R12,12(R13) SAVE REGISTERS 14 THRU 12
         L     R13,8(R13)     SET 13 TO NEXT IMS PRE-CHAINED SAVE SET
         LR    R12,R15        SET 12 AS BASE
         USING INDEXXIT,R12   USE R12 AS BASE FOR PROGRAM
         USING PST,R1         USE R1 AS BASE FOR PST
         USING XRECORD,R2     USE R2 AS BASE FOR INDEX RECORD
         USING DMBXMPRM,R3    USE R3 AS BASE FOR INDEX CSECT
         USING XSOURCE,R4     USE R4 AS BASE FOR INDEX SOURCE SEGMENT
         SPACE 2
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                   *
*  LOGIC SHOULD BE PROVIDED HERE TO DECIDE WHETHER THE INDEX RECORD   *
*  SHOULD BE SUPPRESSED.                                            *
*                                                                   *
*  THE FOLLOWING CODE WILL TEST WHETHER THE KEY OF THE INDEX          *
*  RECORD IS ALL BLANK.  IF THE FIELD IS ALL BLANK, THE INDEX ENTRY   *
*  WILL BE SUPPRESSED.                                              *
*                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         SPACE 1
         CLC   XFIELD1,BLANKS  IS FIELD BLANK
         BE    SUPPRESS        YES, SUPPRESS INDEX FOR FIELD
         B     NOSUPP          NO, ALLOW INDEX FOR FIELD
         SPACE 2
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                   *
* SUPPRESS RETURN, SET 4 IN R15 TO TELL IMS TO SUPPRESS THE ENTRY     *
*                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         SPACE 1
SUPPRESS DS    0H
         L     R13,4(R13)      BACK UP TO PRIOR SAVE AREA
         RETURN (14,12),RC=4   RETURN WITH 4 IN R15
         SPACE 2
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                   *
* NORMAL RETURN, SET 0 IN R15 TO TELL IMS TO NOT SUPPRESS THE INDEX   *
*                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         SPACE 1
NOSUPP   DS    0H
         L     R13,4(R13)      BACK UP TO PRIOR SAVE AREA
         RETURN (14,12),RC=0   RETURN WITH 0 IN R15
         SPACE 2
BLANKS   DC    CL255' '       CONSTANT OF 255 BLANKS
         SPACE 2
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                   *
* GENERATE DSECT FOR THE INDEX RECORD                               *
*                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         SPACE 1
XRECORD  DSECT
XFIELD1  DS    CL5
```

```
         SPACE 2
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                     *
* GENERATE DSECT FOR THE INDEX SOURCE SEGMENT                         *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         SPACE 1
XSOURCE  DSECT                DSECT FOR INDEX SOURCE SEGMENT
XSFIELD1 DS    CL5            FIELD 1 OF INDEX SOURCE SEGMENT
         SPACE 2
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                     *
* DSECT FOR INDEX MAINTENANCE EXIT ROUTINE PARAMETER CSECT             *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         SPACE 1
DMBXMPRM DSECT
DMBXMSGN DS    CL8            NAME OF INDEXED SEGMENT
DMBXMXDN DS    CL8            NAME OF INDEXED FIELD
DMBXMXNM DS    CL8            NAME OF USER EXIT ROUTINE
DMBXMXEP DS    A              EXIT ROUTINE ENTRY POINT ADDRESS
DMBXMPLN DS    H              TOTAL LENGTH OF CSECT
         DS    H              NOT USED
DMBUSERD DS    C              START OF USER DATA IF ANY
         SPACE 2
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                     *
* GENERATE DSECT FOR THE IMS PST WHICH IS PASSED IN R1                *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         SPACE 1
         PRINT NOGEN
         IDLI  PSTBASE=0
         PRINT GEN
         SPACE 2
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                     *
* GENERATE EQUATES FOR SYMBOLIC REGISTERS                              *
*                                                                     *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
         SPACE 1
         REQUATE
         SPACE 2
         END
```

Chapter 9. Segment Edit/Compression Exit Routine (DFSCMPX0)

This chapter describes the Segment Edit/Compression exit routine, its attributes, how to activate it, how the routine communicates with IMS, and the restrictions that apply. The chapter also provides a description of sample Segment Compression/Expansion modules.

In addition, Hardware Data Compression support and its implementation are explained in "Hardware Data Compression Support" on page 104.

In this Chapter:

About This Routine

You can write a Segment Edit/Compression exit routine to compress and expand segments of data. Segment compression saves space and can result in reduced logging. You can write an exit routine to:

- Edit or compress both fixed- and variable-length segments
- Accomplish either data edit/compression (DEDBs or full-function databases) or key edit/compression (full-function databases only).

If you write your own exit routine, you can also allow for editing, such as encoding and decoding segments for security purposes, and for validating and formatting data. The logic for data encoding and decoding (or for other desired editing or formatting) can be based on information contained within the user-written routine itself. It also can be based on information from an external source, such as data provided in the DBD block, or from tables examined at execution time.

Segment compression is possible for either full-function databases or data entry databases (DEDBs). Either routine (DFSCMPX0 or DFSKMPX0) can be used.

Recommendation: Use DFSCMPX0, because it uses MVS services.

You can apply the same exit routine to multiple segment types within the same or different databases.

The Segment Edit/Compression exit routine is optional. No default routine is called. The sample exit routines only perform segment compression and expansion. The exit routines should be implemented by those having overall systems or database responsibility for an installation. These routines should be transparent to the application programs that access the databases.

Related Reading: For a list of the specific full-function databases that are supported and for additional guidance-level information, see *IMS/ESA Administration Guide: Database Manager*.

Attributes of the Routine

The attributes of the Segment Compression/Edit exit routine differ depending upon the type of database that uses the routine.

Full-Function Database
When used with full-function databases, this exit routine has no specific link-editing requirements. For information about guidelines, see "Link-Editing the Routines" on page 5 in "Chapter 1. Guidelines for Writing IMS Exit Routines" on page 3.

The exit routine is serialized if it is called at initialization. When the exit routine is subsequently called, it can depend on any information saved in its control block.

DEDB If the Segment Edit/Compression exit routine is used with DEDBs, it must be written and link-edited as reentrant. In addition, the exit routine is loaded during control region initialization rather than during the opening of a database (as with a full-function database).

Link-Editing the Routine

After an edit routine has been compiled and tested and before it is used by the IMS system, it must be placed into IMS.RESLIB, SYS1.LINKLIB, or into any operating system partitioned data set to which access is provided with a JOBLIB or STEPLIB control region JCL statement. You must also specify one entry point to the exit routine.

Loading the Routine

Each time a database is opened, IMS examines each segment description to determine whether edit/compression has been specified for that segment type. If so, the exit routine is loaded from its resident library by IMS. IMS obtains the name of the routine from the COMPRTN= parameter of the SEGM statement of the DBD.

Related Reading: For details on coding the COMPRTN= parameter, see *IMS/ESA Utilities Reference: System*. Adequate storage for the edit/compression routine must be provided for both batch and online systems.

How the Segment Edit/Compression Facility Works

When a segment requiring editing or compression is accessed, IMS gives your edit routine control and provides it with the following information:

- Address of the data portion of the segment.
- Address of the segment work area.

Definition: Although the exit can be used for functions other than compression, from this point on the use of the term *compression* refers to the process of converting the segment from the application program form to the form written to external storage. The term *expansion* refers to the process of converting the segment from the external storage form to the application program form.

Two types of segments can be presented to the routine: fixed length segments, with a data length that is static and is reflected in control blocks; and variable-length segments, with its data length contained within a field in the first two bytes of the segment itself. While a routine dealing with a single-segment type normally does not need to recognize the differences, a more general purpose module involved with multiple segment types can obtain sufficient information to differentiate between them. This is done by examining data provided in the segment compression control section.

Segments being processed using the segment edit/compression facility are stored as variable-length segments in the database. Variable-length segments have a size field in the first two bytes of the data portion of the segment. This size field defines the length of the data portion of the segment. When segments are defined to the application program as fixed length, your routine must expand it to the fixed length expected by the application program. In reverse, if the application program presents a fixed-length segment, your edit routine must add the size field to the compression segment. If the segment is a variable-length segment, it must update the size field with the correct segment length.

Example

Although your edit routine can modify the key fields in a segment, the segment's position in the database is determined by the original key field.

Example: If the key field of a segment type is based on last names and the database has segments for people named McIvor, Hurd, and Caldwell, these segments are maintained in alphabetic sequence—Caldwell, Hurd, and McIvor. Assume your edit routine encodes the names as follows:

```
Caldwell ------> 29665
Hurd     ------> 16552
McIvor   ------> 24938
```

The encoded value is put in the key field. However, the segments in the database remain in their original sequence (Caldwell, Hurd, McIvor) rather than in the numeric sequence of the encoded values (16552, 24938, 29665). Because segments in the database are maintained in their original sequence, application programs can issue GN calls and retrieve the correct segment even though segments are encoded. This is also true for secondary index fields contained in index source segments.

Using the DBD Table

The DBD control block has a table appended to it in the form of an assembler language CSECT. One CSECT is filled in for each segment type that specifies the use of the segment edit/compression facility. The CSECT contains basic information, such as the name of your edit routine and the name of the segment type. You can extend the CSECT to contain any editing parameters or criteria you want. In other words, some or all of the logic for editing a segment type can be put in the CSECT. You can perform different editing operations on different segment types with a single edit routine. If you do not want the log for editing a segment type in your edit program, any external source can provide it, not just the table in the DBD.

Related Reading: For information on the DBD control statement SEGM, see the section "SEGM Statement" in *IMS/ESA Utilities Reference: Database Manager*.

Activating the Routine

When the application program is activated and begins accessing segments, IMS interfaces with the Segment Compression/Edit exit routine as described in this section. In all cases, IMS passes an entry code to the exit routine. Your exit routine must examine this entry code to determine the function to be performed.

Activating the Routine for Compression

For compression, regardless of the format at the source address, the segment at the destination address must be in variable-length format (see Figure 5). The first data field of the destination segment is a 2-byte segment size field.

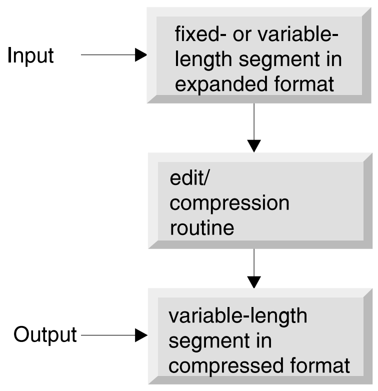

Input
fixed- or variable-length segment in expanded format
edit/ compression routine
Output
variable-length segment in compressed format

Figure 5. Segment Compression

Segment Length

If a fixed- or variable-length segment requires compression and the data format is such that compression cannot take place, the addition of control information by your exit routine, indicating the segment could not be compressed, lengthens the segment beyond the maximum length definition. To allow for this expansion and to allow IMS to check the validity of compression results, you can add an arbitrary value of 10 bytes to the segment length. If the segment length specified in the DBD is variable and the database is a DEDB, the length can exceed the maximum by up to 10 bytes but must not exceed 120 bytes less than the control interval (CI) size. If the segment length specified in the DBD is variable and the database is HIDAM, HISAM, or HDAM, the length must not exceed the DBDGEN-specified maximum.

The length of the segment to be moved is provided in one of two places: If the segment length specified in the DBD is fixed, the source length is in the DMBCPSGL field. If the segment is defined as variable in length, the source length is provided as a binary value in the first two bytes at the source address. In either case, the move operation provided by the edit/compression routine must result in a 2-byte length field, followed by the corresponding quantity of data in the segment work area.

Activating the Routine for Expansion

For expansion, the input segment has a variable-length format as shown in Figure 6.

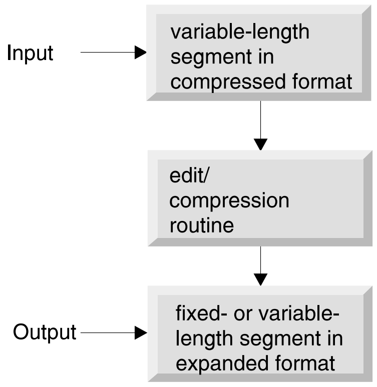

Input
variable-length segment in compressed format
edit/ compression routine
Output
fixed- or variable-length segment in expanded format

Figure 6. Segment Expansion

Entry Code Determination

For segment expansion that occurs during the segment retrieval process, IMS examines the application program request. If the request is satisfied by a compressed segment, a test is made to determine the type of compression used, either key or data. Then, depending upon the type of retrieval request, either entry code 4 or 8 is passed to the expansion routine. The following criteria are used as a basis for the decision:

- If the segment can be accepted without analysis of either a key or data field, control is transferred using entry code 4. The segment is expanded to the form presented to the user.
- If the value of the segment sequence field requires examination prior to segment selection, an additional check is performed to determine data or key compression. Data compression requires no additional processing, while key compression requires activation of entry code 8. If the segment is qualified for presentation after the key field is validated, IMS formats the segment using entry code 4 and passes it to the exit routine.
- If data field analysis is necessary to properly satisfy the DL/I call, proper expansion of the segment by entry code 4 occurs. When the correct segment is found, it is passed to the user.

The format of the segment presented through entry codes 4 and 8 of the compression routine is identical to that of a variable-length segment (a 2-byte segment size field followed by the appropriate quantity of data). The exit routine must expand the segment at the destination address in correct format, either fixed or variable-length. In the case of key compression, the exit routine must expand the segment from its start to the sequence field. For variable-length segments, the segment data length field, after processing by the key expansion, must reflect the length of the expanded portion of the segment at the destination address.

Using IMS Callable Services with This Routine

To use IMS Callable Services with this routine, you must do the following:

- Issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired Callable Service. For details on using the initialization call, see “Step 2. Initializing IMS Callable Services (DFSCSII0)” on page 11.

- Use the PST address found in register 1 as the ECB.
- Link DFSCSI00 with your user exit.

Communicating with IMS

All IMS control blocks provided to the Segment Edit/Compression exit routine are for reference only; no data can be changed, including the segment at the source area address. The only modification allowed is the alteration of the segment during the move operation from the source to the destination address. DSECT addressability to the above-mentioned control blocks is provided by the IMS IDLI macro.

Contents of Registers on Entry

Upon entry to the exit routine, the registers contain the following:

Register	Contents	
0	Set to zero before call to exit routine. May contain Abend code U2990 upon return if the exit routine detected an error.	
1	Address of the Partition Specification Table (PST).	
2	Address of the first byte of the segment to be modified (source address).	
3	Address where the modified segment is returned (destination address). This segment is ten bytes larger than the maximum segment size.	
4	Address of the physical segment description block (PSDB). From this block, the field description blocks (FDB) can be located. (Register 4 is always zero when a DEDB is accessed by the exit routine, because the PSDB does not exist for DEDBs.)	
5	Address of the Segment Edit/Compression control section.	
6	Entry code (detailed in “Description of Entry Codes”):	
	0	Segment compression call
	4	Entire segment expansion call
	8	Partial segment expansion call (full-function databases only)
	12	Full-function database or DEDB area open call
	16	Full-function database or DEDB area close call
13	Address of save area. The exit routine must not change the first three words.	
14	Return address to IMS.	
15	Entry point of exit routine.	

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers.

Description of Entry Codes

The entry code that is passed to the exit routine in register 6 indicates the reason IMS called the exit routine. The five possible entry codes are described in the following sections.

Compressing and Expanding Segments

The following two entry codes are required for segment compression and expansion; they are used when you specify the DATA compression operand.

Code	Description
0	Segment compression call. The source address points to an uncompressed segment image as it appears in the application program input/output area.

Code	Description
4	Entire segment expansion call. The source address points to a compressed segment. Application program requests qualified on a data field require the use of entry code 4 for normal retrieval expansions.

To reduce the amount of processing overhead required with the movement of data, a third entry is required when the KEY compression operand is used. The KEY operand is for use with full-function databases only. Key compression is not supported for DEDBs.

Code	Description
8	Partial segment expansion call with the KEY operand (full-function databases only). Expansion takes place from the start of the segment through the sequence field. This facility is required if you elect to use key compression, or if you compress any field that alters the starting position of the key field. All DL/I calls using sequence field qualification on key compressed segments require the use of this entry code.

Using the Routine with Tabled Data Information

Two additional entry codes are provided allowing two options for processing tabled data information. A Segment Edit/Compression exit routine is called with one of these entry codes when the INIT operand is specified on the COMPRTN= parameter of the SEGM statement.

With these codes, control is passed to the initialization and termination subroutines immediately after the full-function database or DEDB area is opened and immediately before the full-function database or DEDB area is closed. Any processing required for the database segments that cannot be directly related to any one segment can be done at this time using these options. (Initialization processing and termination processing can include the loading and deleting of the compression algorithm table.)

The first option is contained within the DBD module itself. For each segment defined during DBDGEN as eligible for edit/compression, an entry is developed in an assembly language control section, described in “Description of Parameters” on page 100. This control section can be extended by assembling and link-editing it to contain any desired data or algorithm information.

Code	Description
12	Initialization processing call. Control is obtained for algorithm initialization processing immediately after the full-function database or DEDB area is opened. Registers 2 and 3 are unpredictable.

The second option allows the module to issue IMS Callable Services to provide functions equivalent to the LOAD/DELETE or GETMAIN/FREEMAIN macro instructions. They bring additional information into storage in the form of modules from IMS.RESLIB. An example of this is a table of substitution characters maintained separately from the executable code. This table can reflect different combinations for different segments, resulting in a general purpose, table-driven routine capable of processing several segment types.

Code	Description
16	Termination processing call. Control is obtained for algorithm termination processing immediately before the full-function database or DEDB area is closed. Registers 2 and 3 are unpredictable.

When control is passed to the exit routine as a result of the above two entry codes, execution is not in cross-memory mode. Execution is in the control region address space or, if an IMS separate address space is used (LSO=S), execution is in the IMS address space.

Description of Parameters

To help you provide parameters to the edit/compression routine, the DBD control block has a table appended to it that is made up of assembly language control sections. One control section is developed for each segment type to be edited or compressed. Each control section has a CSECT name equal to that of the segment name.

These control sections are placed at the end of the DBD module. They contain information such as the segment edit/compression routine name, the name of the segment, and the total length of that control section. Each control section can be extended to contain any desired data or algorithm information. A sample segment control section is shown in Figure 7.
Information in the various fields shown in Figure 7 is as follows:

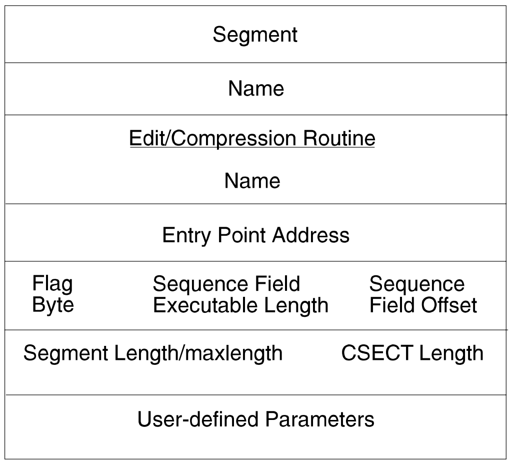


Figure 7. Segment Edit/Compression Control Section (SEGPAC)

```
DMBCPAC    DSECT
DMBCPCNM   DS      CL8     Segment name
DMBCPCSG   DS      CL8     Edit/Compression routine name
DMBCPEP    DS      A       Entry point address
DMBCPFLG   DS      XL1     Flag byte
DMBCPKEY   EQU     X'02'   Segment has key compression
                           option
DMBCPNIT   EQU     X'01'   Initialization processing is
                           required
DMBCPVLR   EQU     X'04'   Segment is variable-length
DMBCPSEQ   EQU     X'08'   Segment has key sequence field
                           defined
DMBCPJJD   EQU     X'10'   Exit caller requests a return code
                           rather than hard abending.
DMBCPSQF   DS      XL1     Executable length of sequence
                           field, if defined
DMBCPSQL   DS      H       Sequence field offset
DMBCPSGL   DS      H       For fixed length segments -
                           segment length; for variable
                           length segments - maximum
                           length
```

```
DMBCPLNG   DS      H       Total length of CSECT; fixed
                           length plus length of
                           user-defined parameters (always
                           a multiple of 8)
DMBCPUSR   DS      0F      Any quantity of user-defined
                           data.
```

The first 28 bytes are constants defined by DBDGEN. When the new table is defined to include additional parameters, these fields must be duplicated. The only exception to this rule is that the CSECT length field must be updated to reflect the new length. After an assembly of the new table, a link-edit is done to exchange the new table for the old one. User-added code should not contain address constants, because this CSECT is moved after it is loaded. Use an ENTRY statement to specify the name of the DBD when this operation takes place, as well as an ORDER statement to ensure that the original order of multiple CSECTs is maintained. For details about this, see the section on automatic CSECT replacement in the MVS product library.

If your exit routine references IMS control blocks other than the one shown in Figure 7 on page 100, you need to reassemble the routine using the current release of IMS.

Restrictions

Keep the following in mind when using the Segment Edit/Compression Facility:

- Because this routine becomes a part of the IMS control or batch region, any abnormal termination of this routine terminates the entire IMS region.
- The exit routine cannot use operating system macros such as LOAD, GETMAIN, SPIE, or STAE.
- All editing or compression of segments occurs as the segments are described in a physical database only. For specific restrictions, see *IMS/ESA Administration Guide: Database Manager*.

Description of Sample Segment Compression/Expansion Modules

Compression/expansion examples are provided as guidance to the IMS system user. The examples are not intended to be operational (for example, they contain many unspecified series of routines), and no support by IBM for these routines is implied. These programs process a particular segment for compression or expansion on the basis of the parameters and data passed by the IMS Control Program.

DFSCMPX0 and DFSKMPX0 can be used by either full-function databases or DEDBs. Both routines perform segment compression. The only differences are:

- DFSCMPX0 compresses three or more repeated strings. This exit routine employs MVS services to accomplish segment compression and expansion. For more information on these services, see the MVS/ESA library. (DFSCMPX0 is the recommended compression routine.)
- DFSKMPX0 compresses four or more repeated strings. This exit routine relies on programming logic to accomplish segment compression and expansion. (DFSKMPX0 is not recommended, but it will continue to be supplied and supported for compatibility reasons.)

When control is given to DFSCMPX0 or DFSKMPX0, the program checks the entry code passed in register 6. The entry code indicates whether the request is for compression of a segment or for the partial (full-function databases only) or entire expansion of a compressed segment. It then branches to an appropriate routine to perform the required task. Upon normal completion of the task, it returns control to the IMS Control Program with a return code of 0.

Specific rules and restrictions followed in compression and expansion of a segment are detailed in the following sections. For sample code, see the IMS.DBSOURCE library.

The Compression Routine

Compression of a segment requires different data handling according to the data organization of the segment. The two data formats are:

Fixed data

Variable-length data

A user can specify the KEY (full-function databases only) or DATA operand for either of the two data formats.

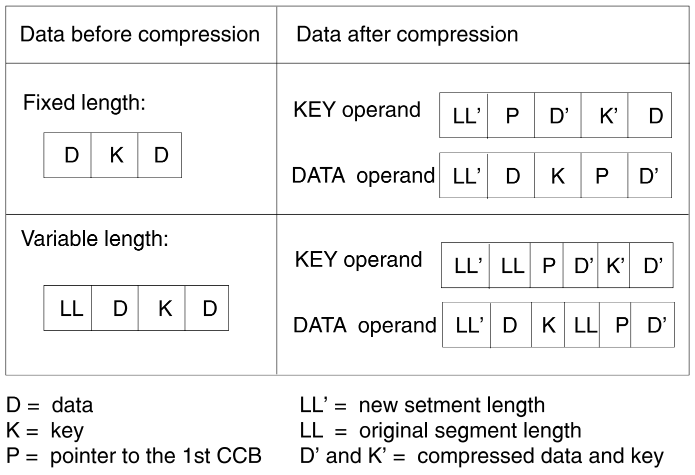


Figure 8. Data Handling Formats

Compression of a segment results in one of the four formats listed in Figure 8, depending upon the original record format and the operand specified.

The Initialization Processing Routine

When specified, IMS gives control to the segment edit/compression routine immediately after the databases are opened and immediately before the databases are closed.

When a command code is given to branch to the initialization processing routine or to the termination processing routine in the DFSKMPX0 program, the DFSKMPX0 program returns to the calling program. No processing of particular data is attempted at this stage.

Program Messages and Codes

Abend codes

Table 15 lists the abend codes.

Table 15. Program Messages and Codes - Abend Codes

User Abend	Description
2989	A segment data organization is variable-length, but its length field is 2>N>32767 A fixed-length record, but the segment length in Compaction Control Table indicates: 0>N>32767
2990	A command code passed by the control program is out of a valid range: 0>N>16 1. REASON - D4D7E701: During a compression request, the input length of the variable length segment is less than 2 bytes. 2. REASON - D4D7E702: During an expansion request, the input length of the compressed segment is less than 2 bytes. 3. REASON - D4D7E703: During an expansion request, a non-zero return code was returned by the MVS expansion service. (CSRCESRV). 4. REASON - D4D7E704: INIT was not specified in the COMPRTN= parameter of the SEGM statement. 5. REASON - D4D7E705: Invalid function code. A command code passed by the control program is out of valid range. 6. REASON - D4D7E706: The key field length (sequence field) plus the offset of the key field within the segment is greater than the segment length indicated in the segment length field of a Compression Control Table. 7. REASON - D4D7E707: The length of a segment indicated in the segment length field of a Compression Control Table is negative.
2991	A command code is passed to compress after, or expand up to, a sequence field of a segment. No sequence field is defined in the segment.
2992	Any of the following conditions results in an abend with the above code. Applicable to both fixed- and variable-length segments: • A D/K length is greater than an SCL length of a segment. Applicable only to a variable-length segment: • A D/K length is greater than a LL length. • A LL length is greater than an SGL length. • A LL length is less than 2. • An SGL length is less than 2. Applicable to a fixed segment: • An SGL length is a negative value. **D/K length =** A sum of length from the beginning of a segment to the end of a key field (SEQUENCE FIELD). **SGL length =** A length of a segment indicated in the segment length field of a Compression Control Table. **LL length =** A length of a variable-length record indicated in the first two bytes of a precompressed segment.

Program Assumptions

All parameters and data passed by the IMS control program, such as the address of the input segment data, the output data area address, and the length of an input segment, are considered valid data.

The IMS control program passes an address of an input segment data area in register 2 and an address of an output data area in register 3.

The size of output data area is:

- A segment length plus two bytes for a fixed-length segment.
- The maximum segment length for a variable-length segment.
- No segment length greater than 32,767 bytes.

All segments processed by the compression routine are treated as variable-length by the IMS system control program, regardless of their precompression format.

Using the Sample Segment Compression/Expansion Exit Routine

For the latest version of DFSCMPX0, see the IMS.DBSOURCE library; the member name is DFSCMPX0).

DFSKMPX0

Recommendation: It is recommended that you use the DFSCMPX0 sample exit routine.

Although no DFSKMPX0 sample exit routine is provided here, the exit routine is supported and supplied in the IMS.SVSOURCE library.

Hardware Data Compression Support

You can compress or expand full-function and DEDB databases by using Hardware Data Compression support. Hardware Data Compression (HDC) reduces DASD storage requirements for databases, reduces database I/O, and improves database performance.

This section describes the following:

- “How HDC Works”
- “How to Implement HDC Support” on page 105
- “Sample JCL Procedure” on page 107
- “HDC Tips” on page 109
- “Return Codes from the HDCD Utility” on page 109

Restriction: The DEDB Sequential Dependent Scan utility (DBFUMSC0) does not provide any support for SDEP segment decompression. Any decompression would need to be performed against the SDEP scan output data set after DBFUMSC0 terminated successfully.

How HDC Works

HDC enables you to take advantage of the MVS/ESA SP 4.3 Hardware Data Compression special program enhancement (SPE).

With HDC support, you can generate new exit routines to activate the hardware-assisted data compression available on certain ES/9000 processors. The

ES/9000 processors use a compression technique, which uses a fixed number of bits to replace a variable number of bytes.

If compression hardware is installed, the segment is compressed or expanded using the hardware instruction CMPSC. If compression hardware is not installed, the standard HDC exit routine calls the MVS CSRCMPSC macro to compress or expand the segment by activating software simulation.

HDC compresses and expands segment data by calling a compression exit routine that has been specified on the SEGM statement during DBDGEN. This exit routine is created by link-editing a user-defined dictionary and an IMS-supplied base exit routine.

The space saved by compression depends on the user-defined dictionary, which performs the translation between compressed and uncompressed data. The dictionary always has 4,096 entries. The total size of the dictionary is 64 KB, because each entry is 16 bytes (8 for compression and 8 for expansion). Different dictionaries are built for different sets of data. You receive the best results by creating a dictionary that compresses the most frequently occurring data in the largest databases.

If a fixed or variable-length segment requires compression and the data format is such that compression cannot take place, then the exit routine adds control information which indicates that the segment could not be compressed. This addition of the control information will lengthen the segment beyond the maximum length definition. To allow for this expansion and to allow IMS to validity check the compression results, you can add an arbitrary value of 10 bytes to the segment length.

If the segment length specified in the DBD is variable and the database is a DEDB, the length can exceed the maximum by up to 10 bytes but must not exceed 120 bytes less than the control interval (CI) size. If the segment length specified in the DBD is variable and the database is a HIDAM, HISAM, or HDAM, the length cannot exceed the DBDGEN maximum.

The length of the segment to be moved is provided in one of two places:

1. If the segment length specified in the DBD is a fixed length, the source length is in the DMBCPSGL field.
2. If the segment is defined as variable in length, the source length is provided as a binary value in the first two bytes at the source address.

In either case, the move operation provided by the edit/compression routine must result in a 2-byte length field, followed by the corresponding quantity of data in the segment work area.

How to Implement HDC Support

You can use HDC with HISAM, HIDAM, HDAM, and DEDB databases.

To implement hardware data compression, follow these steps:

1. Create an HDC dictionary, using the Hardware Data Compression Dictionary utility (DFSZLDU0).
2. Link-edit the HDC dictionary to an IMS-supplied base exit routine, which produces a segment edit/compression routine. The base module is about 1 KB

and is link-edited with 64-KB dictionaries. Therefore, the user exit routines require slightly more than 64 KB of memory.

3. Specify the newly created segment edit/compression routine in the DBDGEN SEGM statement COMPRTN= parameter.
4. Unload the database using the old DBD.
5. Create the new DBD specifying the new exit routine.
6. Reload the database using the new DBD. (A new DBD requires that you run ACBGEN.)

Building the HDC Dictionary

To build the HDC dictionary, use a sequential variable-length file as input to the HDCD utility. This must be a QSAM file of a variable record format and contain uncompressed segments, which are used to build the dictionary. You can create this QSAM file with a user-written unload program, or with the HD Reorganization Unload utility (DFSURGU0). Use your own data analysis to determine what uncompressed segments to use. Use the QSAM data set with the procedure shown in “Sample JCL Procedure” on page 107.

Exception: If you use a QSAM file created by the DFSURGU0 utility, the dictionary build process includes (will not ignore) the header and trailer records created by the DFSURGU0 utility. Also, the dictionary build process includes (will not ignore) the prefix added to each data segment by the DFSURGU0 utility.

Other HDCD Utility Functions

In addition to creating the HDC dictionary, the HDCD utility provides:

Compression statistics program, which is generated from the QSAM input file or from an alternate file. By using an alternate file, you can compare statistics and evaluate the dictionary’s effectiveness.

The compression statistics program:

- Calculates the potential storage savings percentage as follows:

  SAVINGS=(100-((avg compressed segsize/avg precompressed segsize)*100)).

  If the potential storage savings do not meet the HDCDCTL default parameter’s criteria, a dictionary object file is not built.
- Prints the following statistics:
  - HDCDCTL parameters.
  - Number of segments read.
  - Smallest precompressed and compressed segment sizes.
  - Largest precompressed and compressed segment sizes.
  - Average precompressed and compressed segment sizes.
  - Potential storage savings percentage.

  The value shown for either the smallest or largest uncompressed segment could represent the length of the DFSURGU0 utility header or trailer segment.
- Produces data integrity validation option.
- Produces an object file for the specific HDC dictionary, provided that the following compression criteria are met:
  - Precompressed data matches expanded data if the data integrity validation option is specified.
  - Potential storage savings exceed the user-specified minimum percentage.

Sample JCL Procedure

You must create a QSAM data set that contains uncompressed database segments. Then use the QSAM data set with the following JCL procedure.

```
//HDCDBLD  PROC
//         HDCDNAM=DFSZHDCD,    /*USER SUP. DICT NAME,8 CHARS*/
//         QSAMIN='USER.QSAM',    /* INPUT QSAM FILE NAME    */
//         QSAMIT='USER.QSAMALT', /* ALTERNATE QSAM FILE NAME*/
//         DICTLIB='HDC.DICTLIB', /* DICTIONARY LOAD LIBRARY */
//         DICTNAM='DFSZHXYZ',    /* USER DICT. MEMBER NAME  */
//         CMPXIT='USER.COMPLIB', /* COMPRESSION EXIT LIBRARY*/
//         CMPMBR='CMPXIT01',     /* USER EXIT MEMBER NAME   */
//         RGN=2048K,
//         SYS2=,
//         SOUT=*,
//         UNIT=SYSDA,
//         VOLSER=,


//         CYL=TRK,PRIM=5,SEC=2,BLKSZ=3120
//*********************************************************
//* CREATE STATISTICS AND HDC DICTIONARY OBJECT FILE.      *
//*********************************************************


//HDCDGEN  EXEC PGM=DFSZLDU0,REGION=&RGN,PARM=&DICTNAM
//STEPLIB  DD   DSN=IMS.&SYS2.RESLIB,DISP=SHR
//SYSPRINT DD   SYSOUT=&SOUT
//SYSUDUMP DD   SYSOUT=&SOUT
//HDCDIN   DD   DSN=&QSAMIN,DISP=SHR;
//HDCDIT   DD   DSN=&QSAMIT,DISP=SHR;
//HDCDOUT  DD   DSN=IMS.&HDCDNAM.HDCDOBJ,
//         DISP=(,CATLG,DELETE),
//         UNIT=&UNIT,
//         SPACE=(&CYL,(&PRIM,&SEC),RLSE),
//         DCB=(LRECL=80,BLKSIZE=&BLKSZ,RECFM=FB)
//HDCDCTL  DD  DUMMY    /* 'DUMMY' USES DEFAULT PARMS */
//*
//********************************************************
//* CREATE LOAD MODULE FROM DICTIONARY OBJECT TEXT DECK.  *
//********************************************************


//LINK1    EXEC PGM=IEWL,COND=(0,NE),


//    PARM='SIZE=(180K,20K),RENT,REFR,NCAL,LET,XREF,LIST'
//SYSLMOD  DD  DSN=&DICTLIB(&DICTNAM),DISP=SHR
//SYSUT1   DD  UNIT=&UNIT,DISP=(,DELETE),
//             SPACE=(CYL,(10,1),RLSE)
//SYSPRINT DD  SYSOUT=&SOUT
//SYSLIN   DD  DSN=IMS.&HDCDNAM.HDCDOBJ,DISP=(OLD,DELETE,KEEP)
//*
//********************************************************
//* THE USER COMPRESSION EXIT ROUTINE IS BUILT BY LINKING *
//* MODULE DFSZLDX0 AND THE HDC DICTIONARY TOGETHER. THE  *
//* THE HDC DICTIONARY MUST BE THE FIRST CSECT WITHIN THE *
//* USER EXIT ROUTINE AND ALSO BE ON A PAGE BOUNDARY.     *
//********************************************************
```

```
//LINK2    EXEC PGM=IEWL,


//         PARM='SIZE=(180K,20K),RENT,REFR,NCAL,LET,XREF,LIST'
//SYSLMOD  DD  DSN=&CMPXIT(&CMPMBR),DISP=SHR
<litdata>
//SYSUT1   DD  UNIT=&UNIT,DISP=(,DELETE),
//             SPACE=(CYL,(10,1),RLSE)
//SYSPRINT DD  SYSOUT=&SOUT
//RESLIB   DD  DSN=IMS.&SYS2.RESLIB,DISP=SHR
//DICTLIB  DD  DSN=&DICTLIB,DISP=SHR;
//*********************************************************
//*THE FOLLOWING CONTROL STATEMENTS MUST BE IN THE ORDER AS *
//*  ILLUSTRATED.                                          *
//*                                                        *
//*  DFSZHXYZ: THE HDC DICTIONARY NAME FOR THE SEGMENT.    *
//* (&DICTNAM) THIS HAS TO BE CHANGED TO A FIXED   NAME OF  *
//*           DFSZHDCD SO THAT THE COMPRESSION EXIT DRIVER *
//*           CAN BE LINKED TO IT.                         *
//*                                                        *
//*  DFSZLDX0: THE COMPRESSION EXIT DRIVER ROUTINE.        *
//*                                                        *
//*  &CMPMBR: USER SPECIFIED COMPRESSION/EXPANSION EXIT    *
//*          ROUTINE NAME THAT IS USED ON THE              *
//*          SEGM COMPRTN= (&CMPMBR,DATA) DBD STATEMENT.   *
//*********************************************************
//SYSLIN  DD *
  CHANGE  &DICTNAM(DFSZHDCD)     (&DICTNAM) DICTIONARY NAME
  INCLUDE DICTLIB(&DICTNAM)      DICTIONARY MUST BE 1ST CSECT
  INCLUDE RESLIB(DFSZLDX0)       STANDARD COMPRESSION EXIT
  PAGE    DFSZHDCD
  ENTRY DFSZLDX0
  NAME &CMPMBR(R)                (&CMPMBR) COMPRESSION EXIT
/*
// PEND
```

DD Name Descriptions

HDCDIN DD
The input sequential variable length data set that contains IMS database segment data that you extracted.

HDCDIT DD
The input sequential variable length data set or an alternate file that is used to calculate the compression statistics.

HDCOUT DD
Output HDC dictionary object deck. The MVS format dictionary is built and converted into a link-edit compatible object deck for subsequent use in the dictionary link edit step.

SYSPRINT DD
Compression analysis statistics.

HDCDCTL DD
A data set containing the following control statements:

RECS=
The number of input records to be processed. The default is ALL.

PERC=
The percentage of storage savings to be realized. The default is 5 percent.

INTEG=
By specifying Y or N, this keyword checks or does not check the data integrity of compressed segments. The default is N.

HDC Tips

To decide whether to use HDC, run the HDCD utility and analyze the output statistics to determine how much storage and I/O savings you can achieve.

You may want to limit the use of HDC to one time per database, since its implementation requires an unload and reload of the database.

Recommendation: You should evaluate all of the segments in a database before implementing compression. If you use compression for multiple segment types, implement compression for all of them at the same time.

Because uniquely tailored dictionaries yield the most compression, you should use the dictionaries for high-volume segments to maximize savings.

You can create more generally-tailored dictionaries for other reasons. If you know the type of data in most segments, you can create dictionaries by using a sampling of similar data from many of those segments. For example, you might want general dictionaries for upper-case text, mixed-case text, numeric, alphabetic, and general mixed data. You can use these dictionaries for multiple segment types, eliminating the need to produce unique dictionaries for each segment type.

Compression usually saves I/O for sequential processing and can also save I/O for random processing. Typically, savings for random processing is realized with large database records, especially if the record is spread over multiple blocks or CIs. Compression can reduce the number of blocks or CIs that must be read to access a segment. This is likely to apply to twin chains of multiple blocks or CIs, even after reorganizations.

Return Codes from the HDCD Utility

The following return codes can be issued from the HDCD utility:

Code	Description
0	Utility ended successfully and issued the accompanying DFSZ1170I message.
4	Utility ended successfully and issued the accompanying DFSZ1171W message, but it did not build a dictionary because the requested storage savings percentage was not met.
8	Utility ended successfully and issued the accompanying DFSZ1172E message, but it did not build a dictionary because data integrity checks were detected between a source QSAM input record and its equivalent re-expanded record.
12	Utility ended unsuccessfully and issued the accompanying DFSZ1173W message, because MVS CSRCMPSC is not installed on the machine.
16	Utility ended unsuccessfully and issued the accompanying DFSZ1174E message, because a logic error occurred during invocation of the CSRCMPSC compression service macro.

Related Reading: For more information about these messages, refer to *IMS/ESA Messages and Codes*.

Chapter 10. Sequential Buffering Initialization Exit Routine (DFSSBUX0)

This chapter describes the Sequential Buffering Initialization exit routine. The chapter describes the attributes of the routine, performance considerations, and how the routine communicates with IMS.

In this Chapter:

About This Routine

This exit routine can dynamically control the use of Sequential Buffering (SB) for online and batch IMS subsystems, as well as DBCTL. By using one of the five sample SB routines that IMS provides or one that you write, you can:

- Disallow the use of SB.
- Specify that SB be conditionally activated by default whenever IMS detects a sequential I/O pattern in batch or BMP regions.
- Change the IMS default values for the number of buffersets in each SB bufferpool.

The SB exit routine (DFSSBUX0) is called before each application program or utility. This enables the exit routine to dynamically change SB options and parameters and dynamically control how your system uses SB.

Related Reading:

- For more information about SB, see the section on OSAM sequential buffering in *IMS/ESA Administration Guide: Database Manager* .
- For general guidelines on writing DL/I exit routines, see "Chapter 1. Guidelines for Writing IMS Exit Routines" on page 3.

The following sections provide guidelines for writing your own SB exit routine and information about the sample SB routines that IMS supplies.

Table 16 shows the attributes of the Sequential Buffering Initialization exit routine.

Table 16. Sequential Buffering Initialization Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSSBUX0.
Link editing	Follow the guidelines described in "Link-Editing the Routines" on page 5. After you compile and test your module, link-edit it into IMS.RESLIB, SYS1.LINKLIB, or into any operating system partitioned data set that can be accessed by a JOBLIB or STEPLIB JCL statement for the IMS control, SAS, and batch regions.
Including the routine	No special steps are needed to include this routine.
IMS callable services	This exit is not eligible to use IMS Callable Services.
Sample routine location	See "Using the Sample SB Initialization Routines" on page 114.

Loading the Routine

IMS loads the routine at IMS initialization time.

Considering Performance

DFSSBUX0 is called frequently during the scheduling of MPPs and PSBs of CICS in a DBCTL environment. If you modify an SB sample routine or write your own routine, code it to minimize overhead during the routine's call for these programs.

Communicating with IMS

IMS uses the entry registers, parameter list, and exit registers to communicate with the exit routine.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of parameter list.
10	Address of partition specification table (PST).
11	Address of SCD.
13	Address of save area. The exit routine must not change the first three words.
14	Return address of IMS.
15	Entry point of exit routine.

Description of Parameters

DFSSBUX0 receives the address of a parameter area in Register 1. This parameter area is mapped by the DFSSBUXP macro and contains:

- The region type (batch, BMP, MPP, Fast Path, DBCTL) in the SBPRMREG field.
- The job, program, and PSB names. (Exceptions: IMS utilities executed without a PSB have a DBD name instead of a PSB name.)
- The message classes of the message region (when running in an MPP region).
- The IMS default values for SB options and parameters.

The following paragraphs describe how DFSSBUX0 can change the default values of SB options in the SB parameter area. Each change applies only to the current application program or utility being invoked. The DSECT of the parameter area is presented at the end of the discussion.

Disallowing the Use of SB: The **SBPRMPDI** bit determines whether the use of SB is disallowed. The default value for this bit is *off*. DFSSBUX0 can set this bit *on*, however, to disallow the use of SB and cause IMS to ignore any PSBGEN or SB control card requests to the contrary. You can set this bit during peak periods of online use to save real storage space, especially if your system's real-storage is already constrained.

Conditionally Activating SB by Default: The **SBPRMPAD** bit determines whether IMS conditionally activates SB by default. The default value for this bit is *off*. DFSSBUX0 can set this bit *on*, however, so that IMS samples I/O reference pattern statistics of batch and BMP application programs. If IMS detects both a sequential I/O pattern and a reasonable activity rate, IMS activates SB. This occurs only if PSBGEN and SB control cards provide no specifications to override this process.

Exception: Since statistic sampling has an initialization overhead each time an application program is scheduled, IMS does **not** support conditionally activating SB by default for MPPs, Fast Path regions, or CICS applications.

You may want to use DFSSBUX0 to conditionally activate SB by default in the following situations:

- To activate SB for specific batch and BMP programs and for IMS utilities by setting the bit according to the program, job, or PSB name for a program
- To always set the bit to activate SB for all BMP and batch programs and for utilities for MVS/XA systems that are not storage-constrained
- To set the bit depending on the time of day (for example, during night batch processing when most sequential applications are running and a lot of storage is available for buffering purposes)

Changing the Number of SB Buffer Sets: The **SBPRMPNR** full word field specifies a default value for the number of buffer sets (BUFSETS) in each SB bufferpool. The default value for this field is **4**. However, DFSSBUX0 can set this field to a value ranging from 1 to 25, inclusive. If this value is greater than 1, SB can anticipate the future database calls of a BMP or batch program by concurrently reading the next set of blocks while IMS is processing current database calls.

Recommendation: If your databases are well organized, it is recommended that you set a default BUFSETS value of 2 or 3 to save virtual storage space. If your databases are poorly organized, however, you can set a default BUFSETS value of 6 or greater to increase the chance that what your application program or utility is looking for is already in a buffer set.

DFSSBUX0 can also change the default BUFSETS value based on the time of day. For example, you may want DFSSBUX0 to choose a small value for BUFSETS during daytime main online processing time and a larger value during night batch processing time.

The following DSECT describes the format of the SB parameter area:

```
SBPRMP   DSECT
*
SBPRMP1  EQU   *      *****   READ-ONLY INFO FOR EXIT
SBPRMJOB DC    CL8' '        JOBNAME
SBPRMPGM DC    CL8' '        PGM NAME (BLANK FOR CICS)
SBPRMPSB DC    CL8' '        PSB NAME
SBPRMCLA DC    CL4' '        IMS MESSAGE CLASSES
SBPRMREG DC    X'00'         REGION-TYPE
SBPRMRE1 EQU   1             ...BATCH (EXCLUSIVE CICS)
SBPRMRE2 EQU   2             ...CICS
SBPRMRE3 EQU   3             ...BMP
SBPRMRE4 EQU   4             ...MPP
SBPRMRE5 EQU   5             ...IFP (FAST PATH)
         DC    XL3'00'        RESERVED
*
         DS    0F
SBPRMP2  EQU   *      *****   MODIFIABLE SB PARMS FOR EXIT
SBPRMPNR DC    F'0'           NBR OF BUFFER-SETS
SBPRMPFL DC    X'00'          FLAGS
SBPRMPDI EQU   X'80'          ...DISALLOW USAGE OF SB
SBPRMPAD EQU   X'40'          ...CONDITIONAL SB ACTIVATION BY DEFAULT
*
SBPRMPL  EQU   *-SBPRMP       LENGTH OF PARAMETER AREA
```

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers.

Using the Sample SB Initialization Routines

IMS supplies five SB sample routines (DFSSBU1, DFSSBU2, DFSSBU3, DFSSBU4, and DFSSBU9) for you to use in present form, modify, or use as guidelines for writing your own SB routine. The first module disallows the use of SB; the next four cause IMS to conditionally activate SB by default.

DFSSBU1
The sample Sequential Buffering (SB) exit routine disallows the use of SB.

For the latest version of the DFSSBU1 source code, see the IMS.DBSOURCE library.

DFSSBU2
This sample exit routine causes IMS to activate Sequential Buffering (SB) by default when IMS detects a sequential I/O reference pattern and reasonable activity rate. This exit routine can be used for DataRefresher IMS utilities that can benefit from SB in both batch and BMP regions.

For the latest version of the DFSSBU2 source code, see the IMS.DBSOURCE library.

DFSSBU3
This sample exit routine causes IMS to activate Sequential Buffering (SB) by default when it detects a sequential I/O reference pattern and reasonable activity rate. In batch regions, this applies to all application programs and utilities; in BMP regions, this applies to DataRefresher, as well as those IMS utilities that can benefit from SB.

For the latest version of the DFSSBU3 source code, see the IMS.DBSOURCE library.

DFSSBU4
This sample exit routine causes IMS to activate Sequential Buffering (SB) by default when it detects a sequential I/O reference pattern and reasonable activity rate. This applies to all application programs and utilities in both batch and BMP regions.

For the latest version of the DFSSBU4 source code, see the IMS.DBSOURCE library.

DFSSBU9
This sample exit routine either disallows the use of sequential buffering (SB) or causes IMS to activate SB by default based on specific times of day. The routine is coded as follows:

- The time between 1100 hours and 1400 hours is the peak period for processing online transactions. During this time frame, SB is disallowed.
- During the time between 0900 hours and 1100 hours, and 1400 hours and 1700 hours, SB is neither disallowed nor activated by default for batch and BMP regions.
- The rest of the time, SB is conditionally activated by default for batch and BMP regions.

For the latest version of the DFSSBU9 source code, see the IMS.DBSOURCE library.

Part 3. System Exit Routines

Chapter 11. Type 2 Automated Operator Exit Routine (DFSAOE00)

You can write two types of automated operator (AO) exit routines. The AO exit routine described in this chapter (DFSAOE00) is called a type 2 AO exit routine. It can be used in the DB/DC, DCCTL, and DBCTL environments.

The other AO exit routine (DFSAOUE0) is called a type 1 and can be used in the DB/DC and DCCTL environments. It is described in "Chapter 29. Type 1 Automated Operator Exit Routine (DFSAOUE0)" on page 227.

If both DFSAOUE0 and DFSAOE00 are provided in a DB/DC or DCCTL environment, DFSAOE00 is called first. DFSAOE00 determines which exit routine will process the message, or command.

Related Reading: For an overview and comparison of the two AO exit routines, see *IMS/ESA Operations Guide*.

In this Chapter:

About This Routine

IMS calls DFSAOE00 for:

- IMS system messages destined for the master terminal (the MVS system console in the DBCTL environment)
- Commands entered from a terminal and the responses to those commands
- Commands issued from an AO application using the DL/I ICMD call
- Commands generated internally by IMS

DFSAOE00 intercepts these communications before IMS sends the system message, executes the command, or sends the command response.

With DFSAOE00, you can:

- Modify the text of IMS system messages. DFSAOE00 can also add up to 20 bytes of additional text to the end of a message.

  Any synchronous system "messages" generated by IMS after a command has been entered (such as DFS058 cccc COMMAND IN PROGRESS) are not really messages; they are command responses and cannot be modified.
- Delete IMS system messages. In a DBCTL environment, no trace of the original IMS message to the MVS system console is kept if the message is deleted or modified.
- Direct any message, command, or command response to an AO application.
- Start a BMP job (for example, an AO application). DFSAOE00 can issue SVC 34 to start a BMP. The /START command can override the APARM value on the EXEC statement. The APARM parameter is a way of passing information to the AO application. To retrieve the value specified on the APARM parameter, the AO application can issue the DL/I INQY call.

Table 17 on page 120 shows the attributes for the type 2 AO exit routine.

Table 17. Automated Operator Exit Routine Attributes (DFSAOE00)

Attribute	Description
IMS environments	DB/DC, DBCTL, DCCTL
Naming convention	You must name this exit routine DFSAOE00.
Link editing	Follow the guidelines in “Link-Editing the Routines” on page 5. This exit routine must be reentrant. You must manually link edit the routine with DFSCSI00 to include the routine.
Including the routine	DFSAOE00 is a stand-alone, 31-bit reentrant module that you provide. You specify DFSAOE00 by linking it with DFSCSI00 in the IMS.RESLIB concatenation as a stand-alone module. DFSAOE00 can then be loaded and called. If you link both DFSAOE00 and DFSAOUE0 (the other AO exit routine), both are loaded. DFSAOE00 is called first and can either process the message, or command, or it can return a code indicating DFSAOUE0 should do the processing instead.
IMS callable services	DFSAOE00 can use callable services for storage and AOI functions. It is defined to IMS as a standard exit and receives the callable services token in the standard exit parameter list. This routine does not need to issue an initialization call (DFSCSII0). For guidelines on using callable services, see “Using IMS Callable Services” on page 8. For information on how this exit routine uses AOI callable services see “Using AOI Callable Services”.
Sample routine location	IMS.SVSOURCE

Restrictions

All messages queued to an AO application by DFSAOE00 remain in the IMS subsystem in which they are queued. Only AO applications in the local subsystem can issue a DL/I GMSG call to access the queued messages. Because these messages are not written to a shared queues structure, applications on other IMS subsystems cannot access them.

You cannot use DFSAOE00 to modify or delete commands and command responses. This includes commands from a terminal or an application program, or internally generated commands.

Using AOI Callable Services

DFSAOE00 can use IMS AOI Callable Services to communicate with an AO application. Using AOI Services, DFSAOE00 can pass a message containing one or more message segments to one or more AO applications. AOI callable services functions include:

- INSERT, which inserts a message segment to a message buffer.
- ENQUEUE, which sends a message to one or more AO applications using AOI token names. The message sent on the ENQUEUE request can be a single-segment or multisegment message built with INSERT requests or a single-segment message supplied on the ENQUEUE request. An AO application issues a GMSG call, specifying an AOI token name, to retrieve a message sent from DFSAOE00.
- CANCEL, which removes inserted segments when you decide not to send the message to an AO application.

See “Using IMS Callable Services” on page 8 for guidelines on using callable services.

Types of Messages Passed to This Routine

The following types of messages can be passed to DFSAOE00:

- IMS system messages destined for the master terminal (the MVS system console in the DBCTL environment).
- Commands entered from a terminal, with the following exceptions:
  - `/FORMAT`
  - `/LOOPTEST`
  - `/MSVERIFY`
  - `/RELEASE`
  - `/NRESTART` (sent to the exit routine, however, in the DBCTL environment)
  - `/ERESTART` (sent to the exit routine, however, in the DBCTL environment)

  IMS passes the command after basic edit and optional editing routines have had a chance to modify it. This modified input can contain carriage control characters.
- IMS command responses to the terminal.
- Commands issued from an AO application using the ICMD call.
- Commands generated internally by IMS.

Figure 9 on page 122 shows how processing proceeds when a system message is generated. Figure 10 on page 123 and Figure 11 on page 124 show how processing proceeds when a command is entered at a terminal or from an AO application.

IMS does not pass all system messages, operator-entered commands, and command responses to DFSAOE00. The following are types of messages IMS does **not** pass to DFSAOE00:

- Command responses to an AO application
- Commands issued from an AO application using the CMD call (including the responses to those commands)
- System messages for which the destination is not the master terminal
- Command responses to IMS internally generated commands

Changes the Command Editor Makes

The command editor translates certain control characters in any command you enter from a terminal or in any ICMD call. You need to accommodate this translation when writing your exit routine.

The translation is shown in Table 18.

Table 18. Translation of Control Characters in Commands

From	To
X'14' Restore	X'5D' Right Parenthesis
X'15' New Line	X'40' Blank
X'24' Bypass	X'4D' Left Parenthesis
X'40' Blank	X'40' Blank
X'4B' Period	X'4B' Period
X'4D' Left Parenthesis	X'4D' Left Parenthesis
X'5D' Right Parenthesis	X'5D' Right Parenthesis
X'60' Dash	X'60' Dash
X'6B' Comma	X'6B' Comma

Table 18. Translation of Control Characters in Commands (continued)

From	To
X'6D' Dash	X'40' Blank
X'7E' Equal	X'40' Blank

Commands with Network-Qualified LU Names

If you use network-qualified LU names at your installation, the LU name for LU 6.2 application programs can be 17 bytes long. For IMS commands, the network-qualified LU names must be enclosed in single quotation marks (for example, 'NETID.LUNAME').

If an IMS command with the network-qualified LU name is passed to AO exit routine DFSAOE00, IMS modifies the network-qualified LU name in the input command before the command is passed to the AO exit routine. The single quotes around the network-qualified LU name are replaced with blanks, and the period separating the network-identifier and the LU name is replaced with a colon.

Example:A `/DISPLAY` command with a network-qualified LU name entered at the terminal as:

```
/DISPLAY LUNAME 'NETWORK1.LUNAME1' LUNAME2 INPUT
```

is passed to the AO exit routine or logged to the secondary master as:

```
/DISPLAY LUNAME  NETWORK1:LUNAME1  LUNAME2 INPUT
```

Figure 9 shows processing when a system message is generated.

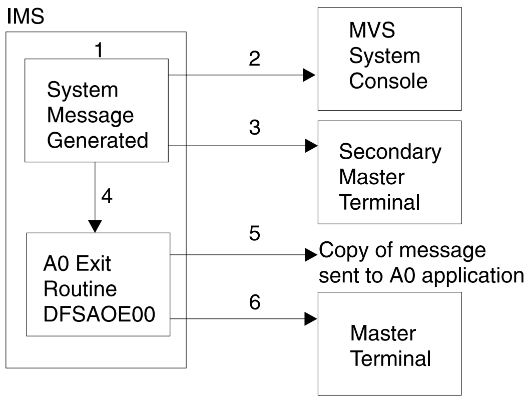


Figure 9. Processing When a System Message Is Generated

Notes:

1. IMS generates a system message destined for the master terminal.
2. A copy of the message may be sent to the MVS system console. This depends on the specific message and is determined by IMS.
3. A copy of the message may be sent to the secondary master terminal if it exists and if you have specified that this is to be done.
4. The copy of the message destined for the master terminal is passed to DFSAOE00.
5. DFSAOE00 can send a copy of the message to an AO application. This is done by enqueuing the message to an AOI token. DFSAOE00 can alter or delete any segment of the message.

6. The message is sent (unless it has been deleted) to the master terminal.

If both DFSAOE00 and DFSAOUE0 had been loaded, this picture would be conceptually the same. However, when DFSAOE00 got control it could either process the message, or it could return a code indicating DFSAOUE0 should be called to do the processing instead.

Figure 10 shows processing when a command is entered at the terminal.

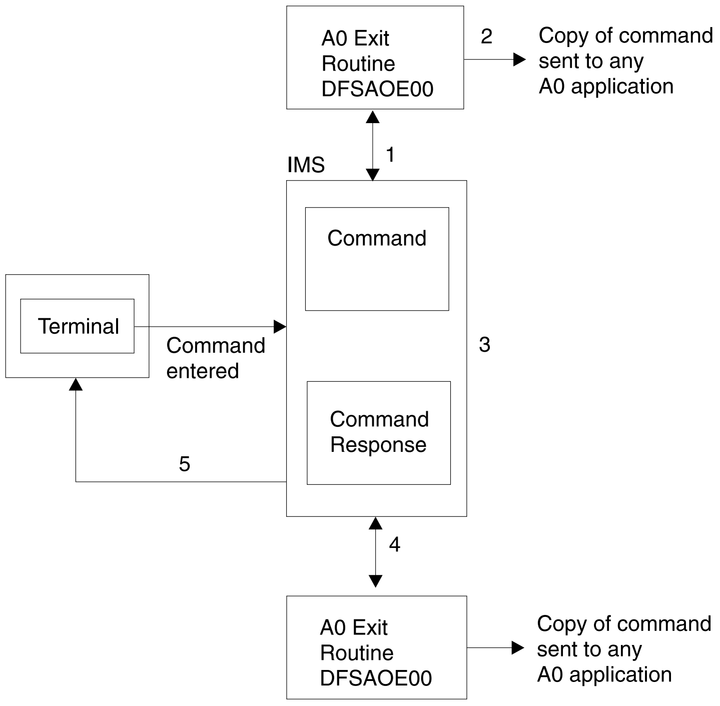


Figure 10. Processing When a Command is Entered at the Terminal

Notes:

1. When a command is entered from a terminal, IMS sends a copy of the command to DFSAOE00 before executing the command.
2. DFSAOE00 can send a copy of the command to any AO application (using the AOI token).
3. IMS executes the command and generates a command response.
4. IMS passes the command response to DFSAOE00. DFSAOE00 can send a copy of the command response to any AO application (using the AOI token).
5. The command response is sent to the terminal that originated the command.

If both AO exit routines (DFSAOE00 and DFSAOUE0) had been loaded, this picture would be conceptually the same. However, when DFSAOE00 got control, it could either process the command or return a code indicating DFSAOUE0 should be called to do the processing instead.

Figure 11 shows processing when a command is entered from an AO application.

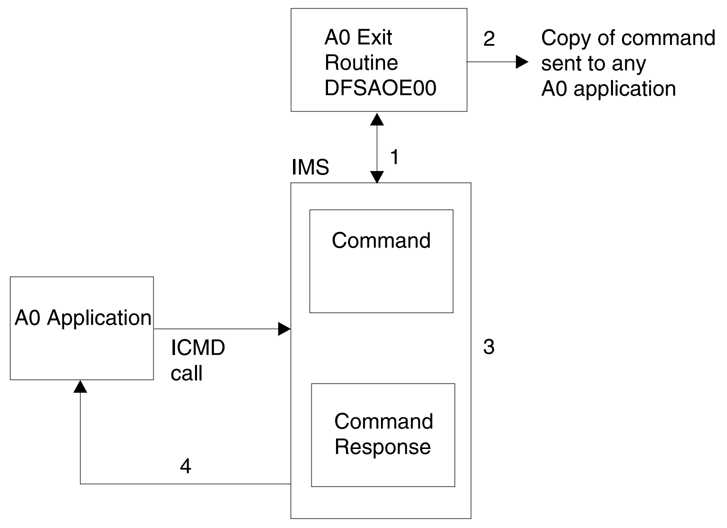


Figure 11. Processing When a Command Is Entered from an AO Application

Notes:

1. When a command is entered from an AO application using an ICMD call, IMS sends a copy of the command to DFSAOE00 before executing the command.
2. DFSAOE00 can send a copy of the command to any AO application (using the AOI token).
3. IMS executes the command and generates a command response.
4. IMS sends the command response back to the AO application.

The type 1 AO exit routine (DFSAOUE0) cannot process commands entered from an AO application.

Activating This Routine

DFSAOE00 is activated:

- During IMS initialization. The purpose of this first entry is so DFSAOE00 can, if you wish, do its own initialization.
- After IMS restart is complete. DFSAOE00 is activated for each system message, command, and command response, as explained in previous sections.

After IMS shutdown processing has begun, DFSAOE00 is disabled and no longer receives control.

Communicating with IMS

IMS communicates with DFSAOE00 through the entry registers and a parameter list.

Content of Registers on Entry

The content of the registers passed from IMS to this exit routine each time it is activated follows:

Registers on Entry

1	Address of the standard exit parameter list.
13	Address of the save area. Your exit routine must not change the first three words of this save area. This save area is **not** chained to any other IMS save area.
14	Return address to IMS.
15	Entry point of this exit routine.

Standard Exit Parameter List

Table 19 shows the content of the standard exit parameter list. When the user exit routine is activated, IMS passes it the address of this list in register 1.

Table 19. Standard Exit Parameter List (Mapped by DFSSXPL)

Field	Offset	Length	Description
SXPLVER	0	4	Address of word containing version number of standard exit parameter list.
SXPLATOK	4	4	Address of word containing callable services token. The callable services token must be included in the callable services parameter list when a callable service is requested.
SXPLAWRK	8	4	Static address of 512-byte work area for DFSAOE00. This is a permanent storage area allocated during IMS initialization and passed to DFSAOE00 each time it is called.
SXPLFSPL	12	4	Address of function-specific parameter list (Table 20 shows this list).
SXPLINTX	16	4	Address of user data table loaded by DFSINTX0 at IMS initialization time. This field is valid in only IMS environments where DFSINTX0 is called. It will be zero in any other environment. For information about the DFSINTX0 exit routine, see “Chapter 36. Initialization Exit Routine (DFSINTX0)” on page 289.

Function-Specific Parameter List

Table 20 shows the contents of the function-specific parameter list. The address of this list is in the standard exit parameter list.

Table 20. Function-Specific Parameter List for DFSAOE00 (Mapped by DFSAOE0)

Field	Offset	Length	Description
AOE0VER	0	4	Address of word containing version number for DFSAOE0.
AOE0FUNC	4	4	Reason for entering DFSAOE00: **1** Initial entry. DFSAOE00 may do initialization functions. **2** Message segment to process. **3** Command is aborted.
AOE0SEG	8	4	Address of message buffer or 0 if this is initial entry. (Table 21 on page 128 shows the message buffer.)

Table 20. Function-Specific Parameter List for DFSAOE00 (Mapped by DFSAOE0) (continued)

Field	Offset	Length	Description
AOE0WRKA	12	4	Address of 256-byte work area used by DFSAOE00. The area is static for the segments of a message, or for a command and the related command responses.
AOE0FLG1	16	1	Entry codes: **X'80'** First segment of multiple segments or first and only segment when X'20' is also on. **X'40'** Middle segment of multiple segments. **X'20'** Last or only segment. **X'10'** Command response will be sent for this command. X'20' is also set when X'10' is set. **X'08'** No segment presented. Last entry to exit.
AOE0FLG2	17	1	Segment or command type: **X'80'** Command entered at terminal. **X'40'** Command response segment. **X'20'** Command (ICMD) issued by AO application. **X'10'** Command generated internally by IMS. **X'08'** IMS system message segment.
AOE0FLG3	18	1	**X'80'** Command input entered at a terminal exceeded 256 bytes.
AOE0FLG4	19	1	Reserved
AOE0DMTK	20	4	Directed message token required to issue AOI callable service requests.
AOE0IMSI	24	8	IMS subsystem identifier.
AOE0IMSL	32	4	IMS version and release.
AOE0SSTY	36	1	**X'01'** DB/DC system **X'02'** DCCTL system **X'03'** DBCTL system
AOE0ROLE	37	1	**X'01'** XRF active **X'02'** XRF alternate **X'03'** RSR active **X'04'** RSR tracker
AOE0MVSL	38	1	MVS version and release on which IMS was generated.

Table 20. Function-Specific Parameter List for DFSAOE00 (Mapped by DFSAOE0) (continued)

Field	Offset	Length	Description
AOE0ENVR	39	1	AO exit routine environment: **1** DFSAOUE0 is loaded. Commands and messages can be passed to DFSAOUE0 to process.
AOE0ORGC	40	4	Origin of the command: **0** Origin fields not set. Fields are set for commands entered from a terminal, an MCS console, LU 6.2 conversation, or OTMA client in a DB/DC or DCCTL system. **1** Origin other than that defined by a specific code **2** VTAM terminal **3** LU 6.2 conversation **4** MCS/E-MCS console **5** OTMA client
AOE0LINE	44	4	Terminal line number (AOE0ORGC=1).
AOE0NODE	44	8	VTAM node name (AOE0ORGC=2).
AOE0NWID	44	8	Network ID (AOE0ORGC=3).
AOE0CONS	44	4	The 4-byte MSC/E-MSC terminal ID (AOE0ORGC=4).
AOE0TMEM	44	16	OTMA member name (AOE0ORGC=5).
AOE0PTRM	48	4	Physical terminal number (AOE0ORGC=1).
AOE0MCSU	48	8	User identification (AOE0ORGC=4).
AOE0LTRM	52	8	Logical terminal name or blanks if LTERM does not exist (AOE0ORGC=1,2).
AOE0LUNM	52	8	Logical unit name (AOE0ORGC=3).
AOE0USID	60	8	Signed-on user ID or blanks (AOE0ORGC=1,2).
AOE0LUUS	60	8	User ID (AOE0ORGC=3).
AOE0TPIP	48	8	Pipe (AOE0ORGC=5).
AOE0USER	68	8	VTAM user, subpool name, or blanks (AOE0ORGC=2).

Table 20. Function-Specific Parameter List for DFSAOE00 (Mapped by DFSAOE0) (continued)

Field	Offset	Length	Description
AOE0RPLY	76	4	Return code from DFSAOE00. This is the only field in this parameter list that DFSAOE00 can modify. **0** DFSAOE00 is not interested in this message or command segment. IMS will process the message as if the exit routine did not exist. Subsequent segments of the message or command response are presented to DFSAOE00. **1** DFSAOE00 is not interested in this message or command segment. Call DFSAOUE0 for processing. Do not call DFSAOE00 for subsequent segments of this message. **2** Send no more segments for this message or command to DFSAOE00. **3** Delete the IMS system message segment. **4** Delete the IMS system message segment and all subsequent segments of this message. **5** DFSAOE00 modified the IMS system message segment.

Message Buffer

The message buffer contains an IMS message, command, or command response. Table 21 the format of the message buffer. The function-specific parameter list points to this buffer.

Table 21. Message Buffer

Field	Length	Description
LL	2	Length of the message on first entry to exit routine. Length includes the length of the LL and ZZ fields but excludes length of the 20-byte work area. (If your exit routine deletes or changes a message or uses the work area, it must update this field.)
ZZ	2	Zeros.
Message text	Variable	Text of the message, command, or command response. **System message:** The first segment of the message text begins with the DFSxxxx number, indicating which message caused IMS to call the exit routine. The message number is followed by the text of the system message. If it is a multisegment message, the remaining segments contain additional text, but do not contain the DFSxxxx message number. **Command:** The message text is one segment long and begins with the delimiter '/', followed by the command. **Command response:** The command response is usually a DFSxxxx message or one segment of a multisegment command.

Table 21. Message Buffer (continued)

Field	Length	Description
CR	1	Carriage return character. This field is optional. If it is there, one byte is included in the LL. Input commands do not include a carriage control character.
Work area	20	Additional area DFSAOE00 can use to add text to the message.

Content of Registers on Exit

There is no requirement for exit registers. DFSAOE00 communicates using the reply field in the function-specific parameter list. DFSAOE00 is passed this list when it is entered.

Register	Contents
14	Return Address
15	0

Chapter 12. Buffer Size Specification Facility (DSPBUFFS)

The Buffer Size Specification facility allows you to control the number of buffers used for RECON data sets when either the local shared resource (LSR) or the nonshared resource (NSR) buffering option is used.

In this Chapter:

About This Facility

DBRC provides a CSECT, DSPBUFFS, for you to override the default number of buffers used. The values in the CSECT are used to build the VSAM local shared resource pool for LSR support or to specify the number of index and data buffers if NSR buffering mode is used.

This facility can be used in DBRC environments.

Link-Editing the CSECT

After assembling the source code, link-edit the object code of the CSECT into the IMS load module DSPCINT0.

DSPBUFFS Layout

The layout of the DSPBUFFS CSECT is shown in Figure 12. You can assemble your own version of this CSECT and replace it in load module DSPCINT0 using the standard linkage editor setup included in the System Modification Program (SMP) process, or modify the existing version of the CSECT supplied by IBM.

```
DSPBUFFS CSECT ,               DECLARE NBR OF INDEX & DATA BUFFERS
         DC    CL8'DSPBUFFS'   REQUIRED EYECATCHER FOR DUMPS
*
*  DECLARE THE NUMBER OF INDEX AND DATA BUFFERS TO BE USED IN EACH
*  OF THE DEFINED OPERATING MODES WHEN USING THE LSR OPTION OF VSAM.
*  APPLIES TO AN ESA* OR XA ENVIRONMENT ONLY. BOTH BUFFER NUMBERS GIVEN
*  IN EACH CASE MUST BE AT LEAST 4 ELSE DBRC REVERTS TO NSR MODE USING
*  THE NSR BUFFER NUMBERS BELOW THAT CORRESPOND TO THE SAME OPERATING
*  MODE.  THIS FEATURE CAN BE USED TO INHIBIT THE USE OF LSR IN ANY OF
*  THE OPERATING MODES SHOULD SOME PROBLEM ARISE.  REMEMBER THAT UNDER
*  LSR THE INDEX/DATA BUFFERS DEFINED APPLY TO ALL THE ACTIVE RECONS.
*
LSRONLIN DC    AL2(6,12)       IMS/ESA ENVIRON-IMS ONLINE DBRC
LSRCICS  DC    AL2(6,12)       IMS/ESA ENVIRON-CICS USE OF DBRC
LSRBATCH DC    AL2(6,12)       IMS/ESA ENVIRON-OFFLINE/BATCH DBRC
```

Figure 12. DSPBUFFS CSECT (Part 1 of 2)

```
*
*  DECLARE THE NUMBER OF INDEX AND DATA BUFFERS TO BE USED IN EACH
*  OF THE DEFINED OPERATING MODES WHEN USING THE NSR OPTION OF VSAM.
*  APPLIES IF THE LSR OPTION HAS BEEN INHIBITED ABOVE FOR ONE OR
*  MORE OF THE DEFINED OPERATING MODES. THE MINIMUM NUMBER OF INDEX
*  AND DATA BUFFERS ASSIGNED TO EACH RECON IS TWO.
*  REMEMBER THAT UNDER NSR THE NUMBER OF INDEX/DATA BUFFERS
*  DEFINED APPLY TO EACH OF THE RECONS.  NOT SHARED AS WITH LSR.
*
NSRONLIN DC    AL2(2,2)        IMS ONLINE DBRC
NSRCICS  DC    AL2(2,2)        CICS USE OF DBRC
NSRBATCH DC    AL2(2,2)        OFFLINE/BATCH DBRC
         END
```

Figure 12. DSPBUFFS CSECT (Part 2 of 2)

As the comments and structure of Figure 12 on page 131 indicate, the first three pairs of halfwords control the number of index and data buffers that are used for LSR. The second three pairs of halfwords control the number of index and data buffers that are used for NSR. DBRC always uses the VSAM LSR option in an ESA or XA environment unless it is inhibited through DSPBUFFS (see comments in the CSECT to see how this is done). In a coexistence environment (see “Example of Specifying Buffers” on page 133), if any release earlier that IMS/ESA V3 is executing in a pre-XA processor, only the NSR section of DSPBUFFS is applicable.

In either LSR or NSR mode, DBRC determines which pair of index/data values to use based on the “operating mode” for each execution. During initialization, DBRC:

1. Uses LSR/NSR pair 1 for IMS control regions
2. Uses LSR/NSR pair 2 when executing under CICS
3. Uses LSR/NSR pair 3 for batch jobs or utilities

In effect, by changing or creating your own version of DSPBUFFS, you can specify separate buffering values for each of the three operating modes. If NSR buffering is used, individual values for BUFNI and BUFND can be specified in the JCL DD statements used to override the default buffer size. For VSAM LSR, only the first three pairs of values are used, so there is no advantage in allocating the RECON data sets through JCL and specifying BUFNI or BUFND values. Similarly, the BUFFERSPACE parameter used when defining a RECON data set through Access Method Services (AMS) is only applicable to the NSR buffering technique and is not used for LSR.

Because the VSAM LSR pools are built while the RECON data sets are open in NSR mode, values for the BUFFERSPACE, BUFNI, and BUFND parameters should not be specified when defining the VSAM clusters and when allocating the RECON data sets using JCL. Because the VSAM LSR pools are built prior to opening the RECON data sets for LSR, supplying values for BUFFERSPACE, BUFNI, or BUFND that exceed VSAM’s minimum default only increases the virtual storage needed to support DBRC for batch and CICS regions.

Use DSPBUFFS to specify the number of buffers for NSR, even though it is optional. With NSR specified, more efficient use of virtual storage can be achieved than by using the BUFFERSPACE parameter (when defining the RECON clusters) and adjusting the number of index and data buffers through the use of JCL. As a result, the RECON data sets can be dynamically allocated in nearly all applications.

Using IMS Callable Services with This Routine

IMS Callable Services are not applicable for use with this exit routine.

Example of Specifying Buffers

Company XYZ shares RECON data sets between two processors. Processor A is an ESA machine, processor B is not— a coexistence environment involving an earlier release of IMS is on processor B. In this case, each IMS system uses a separate copy of the DSPBUFFS example shown below.

XYZ frequently runs batch jobs using DBRC under TSO. However, tight region restrictions exist for jobs run under TSO, so they must limit the amount of storage used by DBRC in these circumstances. On the other hand, DBRC storage is not limited when executing as a control region task, so they have replaced DSPBUFFS with the following values:

```
DSPBUFFS CSECT ,                   DECLARE NBR OF INDEX & DATA BUFFER
         DC    CL8'DSPBUFFS'       REQUIRED EYECATCHER FOR DUMPS
*
*                                  processor A (LSR) SETUP
LSRONLIN DC    AL2(10,26)          ESA ENVIRON - IMS ONLINE DBRC
LSRCICS  DC    AL2(6,12)           ESA ENVIRON - CICS USE OF DBRC
LSRBATCH DC    AL2(6,14)           ESA ENVIRON - OFFLINE/BATCH DBRC
*
*                                  processor B (NSR) SETUP
NSRONLIN DC    AL2(4,9)            NONESA ENVIRON - IMS ONLINE DBRC
NSRCICS  DC    AL2(2,2)            NONESA ENVIRON - CICS USE OF DBRC
NSRBATCH DC    AL2(3,5)            NONESA ENVIRON - OFFLINE/BATCH DBRC
         END
```

When run as an IMS online region, DBRC in processor A (LSR) creates 10 index buffers and 26 data buffers to be shared between the 2 active RECON data sets. In processor B (NSR), DBRC assigns 4 index buffers and 9 data buffers to each RECON data set. When both active RECON data sets are opened for NSR, a total of 8 index and 18 data buffers are implied. Remember that under NSR, when the spare RECON data set is opened, it too will be assigned 4 index and 9 data buffers. For brief periods of time in processor B, the total number of index and data buffers used are 12 and 27, respectively.

Under LSR, when the spare RECON data set is opened (initially in NSR mode, a VSAM requirement), it is assigned 2 index and 2 data buffers. These values cannot be overridden. For brief periods of time in processor A, the total number of index and data buffers used are 12 and 28, respectively. Thus the total amount of storage that is used for RECON buffers is approximately the same in both processors.

When running batch jobs, DBRC in processor A creates 6 index buffers and 14 data buffers to be shared between the 2 active RECON data sets. In processor B, DBRC assigns 3 index buffers and 5 data buffers to each RECON data set opened with NSR buffering. Again, during those periods of time that all 3 RECON data sets are open, the total amount of buffer storage used is approximately the same in both processors (8 index and 16 data buffers in processor A, 9 index and 15 data buffers in processor B).

Chapter 13. Command Authorization Exit Routine (DFSCCMD0)

The Command Authorization exit routine (DFSCCMD0) can be used to verify that a command is valid from a particular origin. DFSCCMD0 is an optional exit routine for commands entered from IMS terminals, including LU 6.2 and OTMA.

DFSCCMD0 is a required exit routine if it is specified to authorize commands entered from:

- ICMD DL/I calls (from automated operator applications)
- MVS MCS or E-MCS consoles

This exit routine verifies that the user is authorized to issue a particular command. IMS does not call this exit routine for internally generated or auto-restart commands.

In this Chapter:

About This Routine

You can use the Command Authorization exit routine with a security product, such as the Security Maintenance Utility (SMU) or the Resource Access Control Facility (RACF). The return code that the exit routine issues ultimately determines the success or failure of the command authorization; the exit routine can override the outcome of either SMU or RACF.

The Command Authorization exit routine is optional. For the latest version of DFSCCMD0 see the IMS.SVSOURCE library; the member name is DFSCCMD0. This sample includes routines for terminals defined using the Extended Terminal Option (ETO) feature, commands entered with ICMD calls, and commands entered from MCS/E-MCS consoles.

Related Reading: For more information on LU 6.2 and ETO, see *IMS/ESA Administration Guide: Transaction Manager*.

Table 22 shows the attributes for the Command Authorization exit routine.

Table 22. Command Authorization Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DBCTL, DCCTL
Naming convention	You must name this exit routine DFSCCMD0.
Link editing	You can assemble the sample exit routine or one that you write using the standard IMS macro and copy files and include it in IMS.RESLIB. You must manually link edit this routine with DFSCSI00 to use IMS callable services.

Table 22. Command Authorization Exit Routine Attributes (continued)

Attribute	Description
Including the routine	Include DFSCCMD0 in IMS.RESLIB. This routine is required if one or both of the following parameters is specified in the IMS, DBC, or DCC procedures: • AOIS=A or C • CMDMCS=B or C If you specify one of these parameters and do not include DFSCCMD0 in IMS.RESLIB, IMS system initialization ends with a U0718 abend. Otherwise, the routine is optional.
IMS callable services	This exit routine can use callable storage services. DFSCCMD0 is defined to IMS as a standard user exit. Exit routines that are defined to IMS receive the callable service token in the standard exit parameter list. This exit routine must issue an initialization call (DFSCSII0) to use callable services and must be manually link-edited with DFSCSI00. For guidelines on using callable services, see "Using IMS Callable Services" on page 8.
Sample routine location	IMS.SVSOURCE

Using the Routine With AO Applications that Issue the ICMD Call

The Command Authorization exit routine can be used with automated operator (AO) applications that issue the ICMD call. The routine is called for AO applications when the AOIS parameter is specified as A or C in the IMS, DBC, or DCC procedure.

DFSCCMD0 is called during ICMD processing to check that the AO application was authorized to issue the command that it issued. DFSCCMD0 lets you secure commands issued in the ICMD call at the command verb, keyword, and resource name level.

Using the Routine With LU 6.2 Application Programs

When an IMS command is received from an LU 6.2 application program, the Command Authorization exit routine is called. The exit routine is called after a RACF (or equivalent) call is made, regardless of the result of the RACF security check. If neither RACF or the Command Authorization exit routine is available to authorize the command, a default level of command security is provided by IMS for commands from LU 6.2 application programs. The commands included in the default are `/BROADCAST`, `/LOG`, and `/RDISPLAY`.

Using the Routine with Static Terminals

The Command Authorization exit routine can be used with terminals defined statically at system definition. If SMU is requested, it performs the command authorization. If SMU is not requested, default terminal security checks the authorization. The return code from SMU or default terminal security is passed to the Command Authorization exit routine. IMS calls the exit routine (if it is included in the system) regardless of the result of the SMU or the default terminal security check; the return code from the exit routine determines authorization.

Related Reading: For more information on default terminal security, see *IMS/ESA Administration Guide: System*.

Using the Routine with ETO Terminals

The Command Authorization exit routine can be used with terminals that are defined dynamically using ETO. If RACF (or an equivalent security product) is requested and the user is signed on, RACF performs the command authorization. IMS passes the RACF return code to the Command Authorization exit routine. IMS calls the exit routine (if it is included in the system) regardless of the result of the RACF security check.

If RACF is not requested but the Command Authorization exit routine is included in the system, IMS calls the exit routine and performs command authorization only. If neither RACF nor the Command Authorization exit routine is included, IMS provides command authorization equivalent to the default terminal security available for static terminals.

The `/SIGN` and `/RCLSDST` commands are the only commands that can be entered from an ETO terminal before sign-on. Although these commands cause IMS to call the Command Authorization exit routine, neither RACF nor the exit routine authorizes the commands.

Using the Routine with Commands from MCS/E-MCS Consoles

This exit routine can be used with commands entered from MCS/E-MCS consoles. The routine is called for commands from MCS/E-MCS consoles when the CMDMCS parameter is specified as B or C in the IMS, DBC, or DCC procedure.

DFSCCMD0 is called during command processing to check that the terminal is authorized to issue the command. DFSCCMD0 lets you secure commands at the command verb, keyword, and resource levels.

Using the Routine With IMS Open Transaction Manager Access

The Command Authorization exit routine can be used with IMS Open Transaction Manager Access (OTMA).

Using the Routine in a Shared Queues Environment

When running in a non-shared queues environment, the name in the field CNTNAME1 of the CNT representing the WTOR LTERM will be WTOR. In a shared queues environment, the name in field CTNAME1 of the CNT representing the WTOR LTERM will be IMSID if it is running in a non-XRF environment and RSENAME if it is running in an XRF environment.

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the exit routine.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of the standard exit parameter list.

Register	Contents
13	Address of the save area. Your exit routine must not change the first three words of this save area.
14	Return address of IMS.
15	Entry point address of exit routine.

Table 23 shows the standard exit parameter list.

Table 23. Standard Exit Parameter List (Version 1, Mapped by DFSSXPL)

Field	Offset	Length	Description
SXPLVER	0	4	Address of word containing version number of standard exit parameter list
SXPLATOK	4	4	Reserved
SXPLAWRK	8	4	Reserved
SXPLFSPL	12	4	Address of function-specific parameter list

Table 24 shows the function-specific parameter list.

Table 24. Function-Specific Parameter List for Command Authorization Exit

Offset	Length	Description
0	4	ECB address (CLB or PST).
4	4	SCD address.
8	4	Address of user table loaded in DFSINTX0.
12	4	For an LU 6.2 conversation, the address of the user ID associated with the command. For an ICMD call, the address of either the user ID or the program name, depending on the region type and whether a GU call was issued. The following list shows the contents of this field according to region type. **Region** — **Parameter Contents** BMP — The user ID in the BMP JCL (USER=). DBCTL — A security token passed in the parse parameter list (PAPL). IFP (without GU) — The program name associated with the command. IFP (with GU) — The user ID or LTERM name (not signed on) for the terminal where the command was issued. Message-passing BMP (without GU) — The program name associated with the command. Message-passing BMP (with GU) — The user ID or LTERM name (not signed on) for the terminal where the command was issued.
16	4	Address of CVB entry for command.

Table 24. Function-Specific Parameter List for Command Authorization Exit (continued)

20	4	Security return code (from SMU, RACF, or equivalent) or decoded security return code (X'80000000'=RACF not called). For request type=3 or 4, a decoded security return code that has one of the following values: X'80000000'= RACF not called X'00000000'= user authorized to RACF to issue command X'00000004'= RACF not available X'00000008'= user ID not defined to RACF X'0000000C'= command not protected by RACF X'00000010'= user ID not authorized to issue command
24	4	CTB address. The value of this parameter is zero for an ICMD call. The parameter contains limited fields for LU6.2 or OTMA.
28	4	Address of input command buffer. For commands from an LU 6.2 conversation this field contains only the first message segment (used for subcommand differentiation.)
32	1	Indicator for contents of user ID field: **U** user ID **L** LTERM **P** PSB name **O** Other name
33	1	Reserved.
34	1	RACROUTE call type (applicable if request type 3, 4, or 6) has one of the following values 0= RACF not called 1= RACF VERIFY call 2= RACROUTE FASTAUTH call
35	1	Type has one of the following values: 1= Terminal security authorization (SMU or RACF) 2= LU 6.2 terminal security authorization 3= ICMD security for user ID 4= ICMD security for program 5= OTMA security for program 6= MSC/E-MCS command security
36	4	Address of three-word parameter list containing RACF security return codes (only included if request type=3, 4, or 6): 0= SAF return code 4= RACF (or equivalent) return code 8= RACF (or equivalent) reason code For an LU 6.2 conversation, the 17-byte address of the partner LU name.

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers except for registers 15, which contains the return code.

Register	Contents
15	One of the following return codes:

Return Code	Meaning
0	USER/TERMINAL is authorized to use command
4	USER/TERMINAL is not authorized
Negative value	USER/TERMINAL is not authorized. The specified user message is sent to the terminal where command originated.

Chapter 14. Dependent Region Preinitialization Routines

This chapter describes Dependent Region Preinitialization routines. Dependent Region Preinitialization routines enable you to perform any application-unique dependent region initialization.

In this Chapter:

About These Routines

Dependent Region Preinitialization routines can activate any MVS system or data management services for which they are authorized, although they cannot issue DL/I calls or activate IMS system services. Because they receive control after module preload, but before IMS scheduling, you may want to use these routines for such tasks as building an internal table for your applications to access during dependent region processing.

For example, you can use a preinitialization routine to build a table for application decision making. You can maintain this table by using MVS services in the following manner:

- Using MVS storage management services, the preinitialization routine can acquire and format a main storage table.
- Using MVS Name/Token callable services, the preinitialization routine can establish a name/token pair for the storage that provides the user application access to the storage area.

This name/token pair can then be used by the dependent region applications using the Name/Token services to access the table. It is your responsibility to determine what these preinitialization routines do, and how the information is made available to user applications.

Preinitialization routines are not intended to control the IMS dependent region environment. These routines provide installation information that can be shared between applications. This information can be used to control the applications and allow the application to make decisions based on the information in these tables.

The preinitialization routines must not be system-type routines (MVS services, Language, Access Method, etc...) but rather user-written routines.

Related Reading: For guidance-level information to help you decide whether or not you want to write these routines to initialize dependent regions, see "Establishing IMS Security" in *IMS/ESA Administration Guide: System*.

Naming the Routine

Using standard MVS conventions, you can give the routine any name up to eight characters in length. Be sure that the name is unique and does not conflict with the existing members of the data set in which this routine is stored. Because most IMS-supplied routines begin with the prefix "BPE", "CQS", "DFS", "DBF", "DSP", or "DXR", choose a name that does not begin with these letters.

Link-Editing the Routine

Before a dependent region can be initialized, all required dependent region preinitialization routines must be assembled and link-edited into a concatenation of //STEPLIB. Normally, this is IMS.PGMLIB or the associated application program library.

You must link-edit the routine as *reentrant* (RENT).

Activating the Routine

The Dependent Region Preinitialization routines get control **after** the dependent region has IDENTIFIED or SIGNED-ON to the associated Control Region, but **before** IMS scheduling is attempted. These routines execute under the IMS Program Control Task whenever it:

- Is attached or reattached in problem program state/user key 8
- Receives control in the order specified in the PROCLIB member

Each Preinitialization exit routine is identified by an 80-byte record in a DFSINT*xx* member of IMS.PROCLIB, where *xx* is a suffix specified by the PREINIT keyword of the IMS dependent region procedures IMSBATCH, DFSMPR, and IMSFP.

Related Reading: For details about these procedures, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Each record identifies one program in IMS.PGMLIB that is to receive control during dependent region initialization (or reinitialization after an IMS user application program abnormal termination).

The 80-byte record identifying each preinitialization routine is as follows:

Column	Contents
1-8	Routine name and entry point.
9-72	Must remain blank.
73-80	Optional record sequence number.

The routines are given control in the order specified in the member. If a requested routine is not found, the dependent region abnormally terminates with a U0588.

Using IMS Callable Services with this Routine

This exit routine is not eligible to use IMS Callable Services.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the routines.

Contents of Registers on Entry

Upon entry, the routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Zero.
13	Address of save area. The routine must not change the first three words.
14	Return address to IMS.
15	Entry point of routine.

Contents of Registers on Exit

Before returning to IMS, the routine must restore all registers except register 15, which must contain the following:

Register	Contents
15	0

Chapter 15. Dump Override Table (DFSFDOT0)

This chapter describes the Dump Override Table (DFSFDOT0). Use the Dump Override Table to either force or suppress dumps for specified abends.

In this Chapter:

About This Table

The table is loaded during early control region initialization to ensure that it is available any time an abend occurs after initialization. After the table is loaded and initialization completed, IMS ESTAE intercepts each IMS Control Region abend and then calls DFSFDMP0. DFSFDMP0 looks for an entry in the Dump Override Table that matches the abend. If an entry is found, the disposition for the abend overrides both IMS and MVS defaults for dumping. If an entry is not found, or the Dump Override Table cannot be accessed, normal dumping logic is used.

The Dump Override Table suppresses IMS Control Region, IMS DLS Region, and DBRC Region abend dumps only. IMS Dependent region dumps cannot be suppressed with the Dump Override Table.

The only change that the Dump Override Table makes to the dumping process is to force or suppress the initial dump decision. IMS still creates only one dump, even when multiple abends occur and matching entries are found in the Dump Override Table. The Dump Override Table is shipped with a default set of entries. Modify DFSFDOT0 to fit your own needs. After you assemble your customized version, link it into the system to activate the changes.

DFSFDOT Macro

The DFSFDOT macro is an IMS-provided macro that is used to generate the DFSFDOT0 table. A DFSFDOT macro call must be coded for each abend for which you want to force a dump to be taken, and for which you want to suppress a dump. Though this macro is used to build the Dump Override Table, the macro is separate from the Dump Override Table to maintain IMS integrity.

DFSFDOT Parameters and Descriptions

Parameters are required and must be specified when defining the Dump Override Table.

DFSFDOT BEGIN
This parameter is required at the start of the Dump Override Table definition. It must be coded before any other DFSFDOT invocations. When BEGIN is specified, no other options are allowed. If any options are specified, they are ignored.

DFSFDOT END
This parameter is required at the end of the Dump Override Table and must be the last DFSFDOT invocation in DFSFDOT0. When END is specified, no other options are allowed. If any options are specified, they are ignored.

ABEND=
This parameter specifies a user or system abend for which a dump is either to be forced or suppressed. The abend is specified in one of the following forms:

UNNNN, where NNNN is the four-digit decimal number (U0780, U4095) of the abend.

SXXX, where XXX is the three-digit hexadecimal number (S075, S3E7) of the abend.

DUMP=
This parameter specifies whether the abend dump is forced or suppressed. This parameter overrides IMS dump decision logic and the MVS dump request bit. It has two options:

FORCE generates a dump for a non-dumping ABEND. Default = none.

SUPPRESS prevents unwanted dumps. Default = none.

The Dump Override Table can specify an abend code and an action of SUPPRESS. However, IMS cannot suppress all dumps. For example, MVS or another component may write the dump prior to IMS receiving control. In the case of system abend code S122, MVS causes the dump to be written before the abend is issued and before IMS receives control. IMS then issues message DFS3984I stating that the dump has been suppressed. This message is misleading, but as far as IMS is concerned the dump has been suppressed. IMS cannot suppress dumps produced by abends that occur after IMS has already processed the Dump Override Table. In the case of ABENDU0002, IMS has already processed the Dump Override Table.

IMS documentation does not explicitly list every abend that supplies a dump that cannot be suppressed by using the Dump Override Table.

Using IMS Callable Services with This Table

IMS Callable Services are not applicable for use with this table.

A Sample Dump Override Table (DFSFDOT0)

This example shows you how to use the DFSFDOT to construct a Dump Override Table. The table in this example forces dumps for ABENDS S075, U780, and S222; the table suppresses dumps for ABENDS S80A and U790.

```
DFSFDOT BEGIN
DFSFDOT ABEND=S075,DUMP=FORCE
DFSFDOT ABEND=U0780,DUMP=FORCE
DFSFDOT ABEND=S80A,DUMP=SUPPRESS
DFSFDOT ABEND=S222,DUMP=FORCE
DFSFDOT ABEND=U0790,DUMP=SUPPRESS
DFSFDOT END
```

Entries need not be in order.

You can generate a Dump Override Table with no FORCE or SUPPRESS by coding a single DFSFDOT BEGIN/END pair, as follows:

```
DFSFDOT BEGIN
DFSFDOT END
```

Errors

Possible errors include:

ASSEMBLY ERROR
An assembly error is issued when an invalid abend code is specified or conflicting dispositions for an abend code are found.

ABEND U0718
An ABEND U0718 (MODULE LOAD FAILURE) is issued if DFSFDOT0 cannot be loaded.

Messages

DFSFDMP0 issues message DFS3984I when a TCB ABEND code matches an entry in the Dump Override Table. The message appears as:

```
DFS3984I DUMP FOR ABEND _____ FORCED BY DUMP OVERRIDE TABLE
DFS3984I DUMP FOR ABEND _____ SUPPRESSED BY DUMP OVERRIDE TABLE.
```

For the latest version of DFSFDOT0, see the IMS.SVSOURCE library; member name is DFSFDOT0.

Chapter 16. IMS Command Language Modification Facility (DFSCKWD0)

This chapter describes how the IMS Command Language Modification Facility modifies the command keyword table.

Related Reading: For a complete explanation of the IMS command language, see *IMS/ESA Operator's Reference*.

In this Chapter:

About This Facility

Several reasons exist for altering the keyword table. For example, you may want to tailor the keywords and synonyms to satisfy unique requirements. A new keyword or keyword synonym in a new IMS release can conflict with a name already assigned by your installation to a resource such as an LTERM or a transaction. If a new keyword "ABC" is introduced and you already have an LTERM with the name "ABC", you can change the keyword name to "ABCDEFG" and remove the source of the conflict. If the source of the conflict is the new keyword synonym, you can change or delete the synonym.

Another reason you may want to modify the table is to limit the use of the `ALL` parameter for certain keywords by changing the parameter's default value from `ALL=YES` to `ALL=NO` or `ALL=DIS`. Using `ALL=YES` allows the operator to enter IMS commands with the `ALL` option; this requires a significant increase in storage in the IMS general pool and adversely affects IMS performance. To avoid these negative consequences, you can specify `ALL=NO` or `ALL=DIS` to be used with IMS commands except those associated with AOI transactions.

To obtain a listing of the command keyword table, print DFSCKWD0, a member of IMS.SVSOURCE. It contains the IMS keywords and synonyms described in the *IMS/ESA Operator's Reference*.

Details about `ALL=NO` and `ALL=DIS` options and instructions for modifying them are discussed in the following sections.

Changing the Table

Two of the macro statements that appear in the table, KEYWD and SYN, can be replaced to modify the keywords and synonyms. One way of modifying the table is:

1. Edit module DFSCKWD0.
2. Change the KEYWD and SYN macro statements.
3. Reassemble DFSCKWD0.
4. Relink the reassembled DFSCKWD0 in IMS.RESLIB.

Related Reading: *IMS/ESA Operator's Reference* contains the list of reserved words, including command keywords, keyword synonyms, and reserved parameters.

Changes to DFSCKWD0 cannot conflict with the names in this list. Keywords can be changed and keyword synonyms can be added, changed, or deleted, as long as the new keyword or synonym is not a reserved word. For example, a new synonym of “MSDB” for MSDBLOAD cannot be added, because “MSDB” is a reserved parameter. If “MSDB” is made a keyword synonym, the `/DBDUMP DATABASE MSDB` command fails with a syntax error.

KEYWD macro statements must be substituted one-for-one in the table. No new KEYWD macro statements can be added.

KEYWD Macro

KEYWD
keyword,LAST=NO|YES,ALL=YES|NO|DIS

Where *keyword* is the new or changed keyword. `LAST=NO` and `ALL=YES` are the defaults and need not be supplied. `LAST=YES` must be specified if it is the last macro call in the module. A keyword cannot exceed 12 characters in length.

Specifying `ALL=NO` prevents the use of the `ALL` parameter with all IMS commands that apply to the keyword being changed (except for commands issued from AOI programs).

For example, specifying `ALL=NO` for the keyword `LTERM` prevents the use of the `ALL` parameter for the following commands:

```
/BROADCAST LTERM ALL
/DISPLAY ASSIGNMENT LTERM ALL
/DISPLAY LTERM ALL
/LOCK LTERM ALL
/PSTOP LTERM ALL
/PURGE LTERM ALL
/START LTERM ALL
/STOP LTERM ALL
/UNLOCK LTERM ALL
```

Specifying `ALL=DIS` prevents the use of the `ALL` parameter with all `/DISPLAY` commands that apply to the keyword being changed (except for commands issued from AOI programs).

For example, specifying `ALL=DIS` for the keyword `LTERM` prevents the use of the `ALL` parameter for the following commands:

- `/DISPLAY ASSIGNMENT LTERM ALL`
- `/DISPLAY LTERM ALL`

SYN Macro

SYN *synonym*,LAST=YES|NO

Where *synonym* is the desired synonym. `LAST=NO` is the default and need not be specified. `LAST=YES` must be coded if this is the last macro call in the assembly. Synonyms cannot exceed 12 characters in length; they must be defined under the keyword to which they apply.

Error Messages

Any error in a macro statement terminates assembly of the keyword table and results in generation of an error message. The remaining macro statements are error checked, but nothing is generated. All macro assembly errors are severity code 16 errors.

KYTBL001 - SEQUENCE ERROR. XXX CANNOT FOLLOW IKEY
A macro was called which cannot immediately follow an IKEY macro call. XXX is either IKEY or SYN. IKEY calls cannot be modified.

KYTBL002 - XXX CALLED WITHOUT ANY PARAMETER
A macro was called without any parameter. XXX is either IKEY, KEYWD, or SYN.

KYTBL003 - XXX IS NOT A VALID INTERNAL KEYWORD
The parameter specified on an IKEY call (XXX) is not known to the system. IKEY calls cannot be modified.

KYTBL004 - KEYWORD TABLE ASSEMBLY TERMINATED
This message appears as a comment after the first error message in a keyword table assembly. All subsequent macro calls will only perform error checking. No code will be generated.

KYTBL005 - SEQUENCE ERROR. KEYWD MUST FOLLOW AN IKEY CALL
A KEYWD macro was called that does not immediately follow an IKEY call.

KYTBL006 - LENGTH ERROR. XXX TOO LONG
The parameter specified on a KEYWD or SYN macro is more than 12 characters in length.

KYTBL007 - INTERNAL KEYWORD 'XXX' HAS NOT BEEN USED
LAST=YES was specified on either a KEYWD or SYN macro call, but not all internal keywords known to the system have been generated. IKEY calls cannot be modified. LAST=YES must appear only on the last macro call in the assembly.

KYTBL008 - XXX CANNOT BE SPECIFIED AGAIN
Internal keyword 'XXX' has already been used. IKEY macro calls cannot be modified.

KYTBL009 - KEYWD MACRO PARAMETER ALL IS INVALID
ALL=NO was erroneously specified on the KEYWD macro at the time the IMS Command Keyword Table (DFSCKWD0) was modified.

Message DFS058 COMMAND COMPLETED EXCEPT xxx y z... uses the keyword table to replace 'xxx' with the keyword associated with the command that caused the message. Therefore, keywords defined by KEYWD macro calls appear in this message. Other messages, however, are prebuilt, and keywords that may have changed will still appear in these.

Using IMS Callable Services with This Facility

IMS Callable Services are not applicable for use with this facility.

Using the Sample IMS Command Modification Facility

For the latest version of DFSCKWD0, see the IMS.SVSOURCE library; member name is DFSCKWD0.

Chapter 17. Large SYSGEN Sort/Split Input Exit Routine (DFSSS050)

The Large SYSGEN Sort/Split Input exit routine enables you to alter the resource data for user-generated resources. User-generated resources are those that are built as partition data set (PDS) add members during the Stage 1 assembly and are designated for user customization only.

In this Chapter:

About This Routine

The Large SYSGEN Sort/Split Input exit routine uses a control record to process user resources and to call exit routines. Each user-generated resource type must have a control record created for the Resource Information File during Stage 1. The control record instructs the Sort/Split utility to process a user resource and to call the Sort/Split utility exit routines. The details of the control record layout are provided in the prolog of the LGNBLKS macro.

DFSSS050 is called by the Large SYSGEN Sort/Split routine following the reading of each user input resource and before the sort table is built.

Related Reading: For more information about the Large SYSGEN Sort/Split routine, see *IMS/ESA Administration Guide: System*.

DFSSS050 allows you to customize the resource data via a return code to the calling routine in register 15. You can insert, delete, or update the resource data.

Attributes of the Routine

Table 25 shows the attributes of the Large SYSGEN Sort/Split Input exit routine.

Table 25. Large SYSGEN Sort/Split Input Exit Routine Attributes

Attribute	Description
IMS environments	DB,DC, DCCTL
Naming convention	You must name this exit routine DFSSS050.
Link editing	Follow the guidelines described in "Link-Editing the Routines" on page 5.
Including the routine	No special steps are needed to include this routine.
IMS callable services	IMS Callable Services are not applicable for use with this exit routine.
Sample routine location	IMS.SVSOURCE (member name DFSSS050)

Restrictions

Only one function can be performed per call.

If a return code other than those listed is returned to the calling routine, an error message is issued, and further processing ceases.

The length of the input buffer and the user buffer must be the same.

The length of the buffers cannot be altered or unpredictable results occur. The buffer length is equivalent to the length of the resource data in 80-byte increments, plus 10 bytes for the sort key.

This routine cannot be used in a DBCTL environment.

Communicating with IMS

The input buffer pointer, input buffer length, and user buffer pointer are passed to DFSSS050 on each call. The input buffer pointer and input buffer length (used for updates only) pertain to the buffer containing the current resource data.

The input buffer contains a “complete” resource. If the resource is a single record resource, the input buffer is 90 bytes in length, where:

- The first 10 bytes are the sort key.
- The following 80 bytes are the actual resource data.

If the resource comprises multiple records, the length of the input buffer is large enough to contain all the 80-byte increments for the resource, plus the 10-byte sort key. The sort key is in the first 10 bytes of the buffer. The resource records follow in 80-byte increments.

The user buffer pointer contains the address of the buffer where an insert is made. The length of the user buffer is the same as the input buffer.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of the input parameter list (three words long).
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of exit routine.

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers except register 15, which must contain one of the following return codes:

Return Code	Meaning
X'10'	Return Next: You do not want to modify the current resource, but would like to be called with the next resource.
X'20'	Insert: You want to make an insert into the data of the resource preceding the one currently in the input buffer. A pointer to the user buffer is provided for the insert. Once the insert is complete, the user buffer is cleared, and DFSSS050 is called again, with the current resource still in the input buffer. Subsequent inserts can be performed before processing the resource currently in the input buffer.
X'40'	Delete: You want to delete the resource currently in the input buffer. The return code of X'40' is all that is needed to delete the current resource.

X'60'	Update: You want to update the resource in the input buffer. The change is made to the input buffer, and then control is returned to the calling routine.
X'90'	Do not call: You want to process the remaining resources for this user type as a non-user type resource. After this return code is issued, DFSSS050 is not called again for this user resource type and prior to the sort step.

Altering the sort key can affect the sorting process of the resource type.

Chapter 18. Large SYSGEN Sort/Split Output Exit Routine (DFSSS060)

The Large SYSGEN Sort/Split Output exit routine enables you to alter the resource data for user-generated resources. User-generated resources are those resources that are built as partition data set (PDS) add members during the Stage 1 assembly and are designated for user customization only.

In this Chapter:

About This Routine

The Large SYSGEN Sort/Split Output exit routine uses a control record to process user resources and to call exit routines. Each user-generated resource type must have a control record created for the Resource Information File during Stage 1. The control record instructs the Sort/Split utility to process a user resource and to call the Sort/Split exit routines. The details of the control record layout are provided in the prolog of the LGNBLKS macro.

DFSSS060 allows you to customize the resource data by means of a return code sent to the calling routine in register 15. You can alter the sorted resource at each of the three sections of a member.

DFSSS060 is called by the Large SYSGEN Sort/Split Routine after each user resource type is sorted and before writing each resource for the user resource type.

Related Reading: For more information about the Large SYSGEN Sort/Split Routine, see *IMS/ESA Administration Guide: System*.

Restrictions

There are some restrictions associated with this routine:

- Only one function can be performed per call.
- If a return code other than those listed is returned to the calling routine, an error message is issued, and further processing ceases.
- The length of the input buffer and the user buffer must be the same.
- The length of the buffers cannot be altered, or unpredictable results occur.
- This routine cannot be used in a DBCTL environment.

Using IMS Callable Services with This Routine

IMS Callable Services are not applicable for use with this exit routine.

Communicating with IMS

The input buffer pointer, the input buffer length, the user buffer pointer, and the member section are passed to DFSSS060 on each call. The input buffer pointer and the input buffer length pertain to the buffer containing the current resource; they are used for updates only.

The user buffer pointer contains the address of buffer where an insert is made.

The output PDS contains members for a given resource type. Each member contains three sections where user resource type members can be altered. These sections are:

BOM Beginning of member, where a header record can be inserted

RES Resource section of the member, where a resource can be inserted, updated, or deleted

EOM End of member, where a trailer record can be inserted

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of the input parameter list (four words long).
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of exit routine.

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers except register 15, which must contain one of the following return codes, listed by section of the member:

BOM (beginning of member):

Return Code	Meaning
X'20'	Insert: You want to insert a header record before processing the resource section. The insert is performed from the user buffer. The CSECT record (if the END record exists) and the GEN record (if it exists), are written to the LGENOUT data set. After the insert is processed by the calling routine, this exit is called again with the user buffer cleared.
X'80'	Section done: You do not have any (or any more) header records to insert. This routine is not called again for the BOM section of this member of this user resource type. It is called again at the BOM section of the subsequent members of this resource type.
X'90'	Do not call: You want to process the remaining resources for this user resource type as a non-user resource. This routine is not called again for the remainder of this member or for any subsequent member for this resource type. This is the default request.

RES (resource section of the member):

Return Code	Meaning
X'10'	Return next: You do not want to change the current resource and, would like to be called with the next resource.
X'20'	Insert: You want to insert a resource before processing the current one. The insert is performed from the user buffer. After the insert is processed by the calling routine, this routine is called again with the original resource in the input buffer and the user buffer cleared.
X'40'	Delete: You want to delete this resource. A return code of X'40' is all that is required for the delete.
X'60'	Update: You want to update this resource. The update is performed in the input buffer.
X'80'	Section done: You do not have any (or any more) resources to insert, delete, or update. This routine is not called again for the RES section of this member for this user resource type. It is called again at the BOM section for the subsequent members of this resource type. This is the default request.
X'90'	Do not call: You want to process the remaining resources for this user resource type as a standard resource. This routine is not called again for the remainder of this member or for any subsequent member for this resource type.

EOM (end of member section):

Return Code	Meaning
X'20'	Insert: You want to insert a trailer record before the end record, if the end record exists. The insert is performed from the user buffer. After the insert is processed by the calling routine, this routine is called again with the user buffer cleared.
X'80'	Section done: You do not have any (or any more) trailer records to insert. This routine is not called again for the EOM section of this member for this resource type. It is called again at the BOM section of the subsequent members for this resource type. This is the default request.
X'90'	Do not call: You want to process the remaining resources for this user resource type as a standard (non-user type) resource. This routine is not called again for the remainder of this member or for any subsequent member for this resource type.

The count of resources written per member may be greater or less than the split count, due to the ability of this routine to insert into and delete members. The split count is the number of resources that will assemble in a 4096K-byte region.

Related Reading: Refer to *IMS/ESA Administration Guide: System* for further information.

Using the Sample Large SYSGEN Sort/Split Output Exit Routine

For the latest version of DFSSS060, see the IMS.SVSOURCE library; member name is DFSSS060.

Chapter 19. Log Archive Exit Routine

The Log Archive exit routine produces an edited subset of the complete IMS log.

In this Chapter:

About This Routine

You can use the sample Log Archive exit routine to produce an edited subset of the complete IMS log. The subset log contains the records needed by the IBM Service Level Reporter, (Program Number 5665-397). The IBM Service Level Reporter (SLR) collects statistics about IMS transactions and schedules.

Related Reading: For details on the data collected by SLR, see .

Restriction: The IBM-supplied sample exit routine is applicable only to an IMS DB/TM system and should not be used in a DBCTL environment.

However, a user-written exit routine can run in a DBCTL environment.

Attributes of the Routine

You must write this exit routine in assembler language.

Restrictions

An abend in the exit routine causes the utility to abend. IMS macros cannot be used in the exit routine. Because the performance of the exit routine affects the total performance of the utility, the logic of the exit routine should not be so complicated as to delay the OLDS from being used by the online region.

Using IMS Callable Services with this Routine

This exit routine is not eligible to use IMS Callable Services.

Communicating with IMS

IMS communicates with the Log Archive exit routine through the entry registers, parameter list, and exit registers.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of parameter list.
13	Address of save area. The exit routine must not change the first three words.

Register	Contents
14	Return address to the calling RECON access routine.
15	Entry point of exit routine.

Parameter List

The parameter list contains the following:

Exit Routine Word (Word 1)
This word belongs to the exit routine. On the initialization call entry to the exit routine, this word contains binary 0. The routine can store any value in this word. For example, the word may point to an area allocated for use by the exit routine via the GETMAIN macro. On subsequent calls to the exit routine, this field contains the value left by the routine on its previous invocation.

Call Type Indicator Field Address (Word 2)
Address of a one-byte area containing the call type indicator.

X'01'	Initialization call
X'02'	Log record processing call
X'03'	Termination call

The call type indicator identifies the reason for calling the exit routine, and the exit routine may have a separate routine for each call type. The user exit should not change this field.

Address of Area Containing the Current Input Log Record or Utility Return

Code (Word 3)
The high-order bit of this word is ON to indicate the end of the list. The contents of the remainder of the word depends on the type of call:

- On an initialization call, this word is zero.
- On a log record processing call, this word will have the address of an area containing the current input log record. The first four bytes of the log record are a BSAM RDW (Record Descriptor Word).
- On a termination call, this word will contain the return code for the current utility.

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers except register 15, which must contain one of the following return codes:

X'00'	Active utility continues processing
non-0	Active utility terminates and IMS issues an error message

Termination due to the exit routine on an initialization call or a log record processing call prevents successful execution of the utility. Termination due to the routine on a termination call results in an error message and a nonzero return code, but successful execution is not prevented, because DBRC has already been notified of archive completion.

IMSLOG

The following record types are selected and written to the data set connected to the DD name IMSLOG:

01	Input message.
03	Output message.
06	IMS start/stop.
07	Application accounting (MPP or BMP end).
31	Message queue get unique.
34	Message cancel.
35	Message placed on message queue.
36	Message removed from message queue.
38	Transaction reschedule.
40	Checkpoint records. Only header, trailer, SMB, and CNT block records are written (subtypes 01, 03, 04, and 98 respectively).
42	IMS log header record.

To limit the size of the written log, the message text parts of the 01 and 03 records are truncated to 24 bytes. However, when this truncation occurs, the total length of all message segments is calculated and stored as a negative value in the length field of the first message segment. SLR uses this field to calculate the number of bytes transferred.

The following DCB information applies to the file IMSLOG:

Keyword	Accepted	Default
RECFM	VB	VB
BLKSIZE	6144 or greater	6144
LRECL	6140 or greater	32760 and RECFM=VBS

A program dependent on the sequence numbers of the IMS log records should not be used to process the written log data set.

Sample Log Archive Exit Routine

```
IMSEXIT  CSECT ,
**START OF MODULE SPECIFICATION***************************************
*                                                                    *
*  MODULE-NAME = IMSEXIT                                             *
*                                                                    *
*  DESCRIPTIVE-NAME = SAMPLE IMS ARCHIVE FUNCTION EXIT               *
*                                                                    *
*  COPYRIGHT = NONE                                                  *
*                                                                    *
*                                                                    *
*  FUNCTION:                                                         *
*    WRITES THE RECORDS USED BY SLR V3 (IBM PP, PROG NO 5665-397)    *
*    INTO THE FILE WITH DDNAME IMSLOG. THE FOLLOWING RECORD TYPES    *
*    ARE WRITTEN (ALL IN HEX): 01, 03, 06, 07, 31, 34, 35, 36, 38,   *
*    4001, 4003, 4004, 4098, 42. MESSAGE TEXTS OF 01 AND 03          *
*    RECORDS ARE TRUNCATED TO 24 BYTES.                              *
*                                                                    *
*  LOGIC:                                                            *
*    CASE INIT.                                                      *
*      GETMAIN STORAGE FOR WORK AREAS AND ANCHOR IT IN THE USER      *
*      WORD.                                                         *
*      OPEN OUTPUT FILE.                                             *
*    END CASE INIT.                                                  *
*                                                                    *
*    CASE NORMAL.                                                     *
*      SUBCASE RECORD TYPES 01, 03.                                   *
*        CALCULATE TOTAL LENGTH OF ALL TEXT SEGMENTS.                 *
*        IF (LENGTH OF ALL TEXT SEGMENTS > 24 BYTES) THEN             *
*          TRUNCATE ANY MESSAGE PART TO 24 BYTES.                     *
*          CHANGE SIGN OF TOTAL TEXT LENGTH AND STORE IT BACK AS AN   *
*          INDICATOR.                                                 *
*        ELSE.                                                        *
*                                                                     *
*        COPY RECORD.                                                 *
*      END SUBCASE RECORD TYPES 01, 03.                               *
*                                                                     *
*      SUBCASE RECORD TYPES 06, 07, 31, 34, 36, 38, 42.               *
*        COPY RECORD.                                                 *
*      END SUBCASE RECORD TYPES 06, 07, 31, 34, 36, 38, 42.           *
*                                                                     *
*      SUBCASE RECORD TYPES 4001, 4003, 4004, 4098.                   *
*        COPY RECORD.                                                 *
*      END SUBCASE RECORD TYPES 4001, 4003, 4004, 4098.               *
*    END CASE NORMAL.                                                 *
*                                                                     *
*    CASE TERMINATE.                                                  *
*      CLOSE OUTPUT FILE.                                             *
*      FREEMAIN STORAGE FOR WORK AREAS AND RESET ANCHOR POINTER.      *
*    END CASE TERMINATE.                                              *
*                                                                     *
*  NOTES = SEE BELOW                                                  *
*                                                                     *
*    DEPENDENCIES = NONE                                              *
*                                                                     *
*    RESTRICTIONS = NONE                                              *
*                                                                     *
*    REGISTER CONVENTIONS = SEE LINKAGE                               *
*                                                                     *
*    PATCH LABEL = NONE                                               *
*                                                                     *
*  MODULE-TYPE = PROCEDURE                                            *
*                                                                     *
*    PROCESSOR = ASSEMBLER                                            *
*                                                                     *
*    MODULE-SIZE = SEE ASSEMBLER LISTING                              *
*                                                                     *
*    ATTRIBUTES = REENTRANT                                           *
*                                                                     *
*  ENTRY-POINT = IMSEXIT                                              *
*                                                                     *
*    PURPOSE = SEE FUNCTION                                           *
*                                                                     *
*    LINKAGE = STANDARD OS LINKAGE                                    *
*                                                                     *
*  INPUT:                                                             *
*    REGISTER 1 POINTS TO A 3-WORD PARAMETER LIST:                    *
*                                                                     *
*    USERWORD - PTR(31). CONTAINS ZERO AT INIT CALL, AND A POINTER    *
*               TO A WORKAREA AT NORMAL AND TERM CALLS.               *
*    TYPEPTR  - PTR(31). POINTS TO A 1-BYTE AREA, THAT CONTAINS:      *
*               X'01' - INIT CALL                                     *
*               X'02' - NORMAL CALL                                   *
*               X'03' - TERM CALL                                     *
*    RECPTR   - PTR(31). FOR NORMAL CALL, POINTER TO A LOG RECORD.    *
*                                                                     *
*  FEEDBACK:                                                        *
*    USERWORD - PTR(31). FILLED IN WITH A POINTER TO A GETMAINED    *
*               WORK AREA AT INIT CALL.                             *
*                                                                   *
*  OUTPUT:                                                          *
*    SELECTED LOG RECORDS WRITTEN TO DDNAME IMSLOG                  *
*                                                                   *
*  MESSAGES:                                                        *
*    001 - UNABLE TO GET STORAGE                                    *
*    002 - UNABLE TO OPEN FILE IMSLOG                               *
*    003 - ERROR DURING PUT TO IMSLOG                               *
*    004 - INVALID CALL TYPE                                        *
*                                                                   *
*  ABEND CODES:                                                     *
*    NONE.                                                          *
*                                                                   *
*  EXTERNAL-REFERENCES = NONE                                       *
*                                                                   *
*  ASSEMBLER MACROS:                                                *
*    DCB                                                            *
*    DCBD                                                           *
*    FREEMAIN                                                       *
*    CLOSE                                                          *
*    GETMAIN                                                        *
*    OPEN                                                           *
*    PUT                                                            *
*                                                                   *
*                                                                   *
* NOTES:                                                            *
*   THE FOLLOWING REGISTERS ARE IN THE CODE:                        *
*                                                                   *
*     R6  = RECPTR:  POINTER TO THE INPUT RECORD                    *
*     R9  = PBLDREC: POINTER TO THE RECORD TO WRITE                 *
*     R10 = ENTIND:  ENTRY TYPE                                     *
*                                                                   *
**END OF MODULE SPECIFICATION****************************************
*-------------------------------------------------------------------*
*                                                                   *
*      PROLOG CODE                                                  *
*      - SET UP ADDRESSABILITY                                      *
*      - GETMAIN STORAGE IF INIT CALL                               *
*                                                                   *
*-------------------------------------------------------------------*
         USING *,R15
         B     PROLOG               * BRANCH PAST MODULE ID
         DC    AL1(16)              * MODULE ID LENGTH
         DC    C'IMSEXIT   82.103'  * MODULE ID
         DROP  R15
PROLOG   STM   R14,R12,12(R13)      * SAVE REGS
         LR    R12,R15              * SET NEW BASE REG
         USING IMSEXIT,R12          * SET ADDRESSABILITY
         LR    R11,R1               * SAVE PARM LIST ADDRESS
         L     R1,0(,R1)            * GET WORK AREA POINTER (OR 0)
         L     R7,4(,R11)           * COPY
         SLR   ENTIND,ENTIND        *   ENTRY
         IC    ENTIND,0(,R7)        *     INDICATOR
         LTR   ENTIND,ENTIND        * IS ENTRY TYPE ZERO ?
         BZ    OTHCASE              * YES, SKIP TO ISSUE MSG
         CL    ENTIND,TERMCALL      * IS ENTRY TYPE TOO GREAT ?
         BH    OTHCASE              * YES, SKIP TO ISSUE MSG
         C     ENTIND,INITCALL      * NO, IS THIS INIT ENTRY ?
         BNE   NOGETMAN             * NO, DON'T ISSUE GETMAIN
         L     R0,SIZEWORK          * GET SIZE OF DYNAMIC AREA
         GETMAIN  R,LV=(0)          * GET STORAGE FOR DYNAMIC AREA
         LTR   R15,R15              * REQUEST OK ?
         BZ    GETMOK               * YES, SKIP ON
*                                   * NO, SEND A MESSAGE
         WTO   'IMSE001 - UNABLE TO GET STORAGE',ROUTCDE=11,DESC=7
         B     NOFREEMN             * SKIP TO EPILOG & RETURN
GETMOK   ST    R1,0(,R11)           * SAVE ADDR IN USER WORD
NOGETMAN LTR   R1,R1                * ANY STORAGE GOTTEN ?
         BZ    NOFREEMN             * NO, SKIP TO EPILOG & RETURN
         USING WORKAREA,R1          * SET TEMP LOCATE OF NEW SAVEAREA
         ST    R13,SAVEAREA+4       * SET CHAIN BACK PTR IN NEW SAVEAR
         DROP  R1                   * DROP TEMP LOCATE
         ST    R1,8(,R13)           * SET CHAIN FORWARD PTR IN OLD SAV
         LR    R13,R1               * POINT TO NEW SAVE AREA
         USING WORKAREA,R13         * LOCATE NEW SAVEAREA
         L     RECPTR,8(,R11)       * COPY RECORD POINTER
         SLR   PBLDREC,PBLDREC      * ZERO OUTPUT RECORD POINTER
         C     ENTIND,INITCALL      * IS THIS INITIAL CALL ?
         BNE   NOTINIT              * IF NOT, SKIP ON
*-------------------------------------------------------------------*
*                                                                   *
*      INIT CALL                                                    *
*                                                                   *
*-------------------------------------------------------------------*
         MVC   DYNDCB(LENDCB),LISTDCB * COPY STATIC TO DYNAMIC DCB
         OI    DYNOPEN,X'80'        * SET HIGH ORDER BIT IN OPEN LIST
         LA    R5,DYNDCB            * POINT TO DYNAMIC DCB
         OPEN  ((R5),OUTPUT),MF=(E,DYNOPEN) * OPEN DYNAMIC DCB
         USING IHADCB,R5            * LOCATE DCB
         TM    DCBOFLGS,DCBOFOPN    * OPEN OK ?
         BO    EPILOG               * YES, SKIP TO EPILOG
*                                   * NO, SEND A MESSAGE
         WTO   'IMSE002 - UNABLE TO OPEN FILE IMSLOG',ROUTCDE=11,DESC=7
         L     ENTIND,TERMCALL      * INDICATE TO TERMINATE
         B     EPILOG               * SKIP TO EPILOG AND RETURN
         DROP  R5                   * DROP DCB ADDRESS
*-------------------------------------------------------------------*
*                                                                   *
*      NORMAL CALL                                                  *
*                                                                   *
*-------------------------------------------------------------------*
NOTINIT  C     ENTIND,NORMCALL      * IS THIS NORMAL CALL ?
         BNE   TERMCASE             * IF NOT, SKIP TO TERMINATE CASE
         SLR   R7,R7                * INSERT RECORD TYPE
         IC    R7,RECTYPE(,RECPTR)  * INTO WORK REGISTER
*-------------------------------------------------------------------*
*      BRANCH TO APPROPRIATE RECORD PROCESSING ROUTINE              *
*      VIA BRANCH TABLE                                             *
*-------------------------------------------------------------------*
         C     R7,TYPE01            * RECORD TYPE 01 ?
         BL    RECEND               * NO, SOMETHING LESS, IGNORE IT
         BE    REC0103              * YES, GO PROCESS IT
         C     R7,TYPE42            * NO, RECORD TYPE 42 ?
         BH    RECEND               *   NO, SOMETHING LARGER, IGNORE
         BE    RECCOPY              *   YES, GO PROCESS IT
         BCTR  R7,0                 *   CONVERT RECORD TYPE 02 - 41
         SLL   R7,2                 *     TO A 4-BYTE INDEX
BRANCHTB B     BRANCHTB(R7)         *       USED TO BRANCH IN TABLE
         B     RECEND               * RECORD TYPE 02, NOT USED HERE
         B     REC0103               * RECORD TYPE 03
         DC    2S(X'7F0'(4),RECEND) * RECORD TYPES 04 AND 05, NOT USED
* ABOVE INSTRUCTION IS EQUIVALENT TO 2 BRANCHES TO RECEND
         B     RECCOPY               * RECORD TYPE 06
         B     RECCOPY               * RECORD TYPE 07
         DC    41S(X'7F0'(4),RECEND) * RECORD TYPES 08 - 30, NOT USED
* ABOVE INSTRUCTION IS EQUIVALENT TO 41 BRANCHES TO RECEND
         B     RECCOPY               * RECORD TYPE 31
         DC    2S(X'7F0'(4),RECEND) * RECORD TYPES 32 AND 33, NOT USED
* ABOVE INSTRUCTION IS EQUIVALENT TO 2 BRANCHES TO RECEND
         B     RECCOPY               * RECORD TYPE 34
         B     RECCOPY               * RECORD TYPE 35
         B     RECCOPY               * RECORD TYPE 36
         B     RECEND                * RECORD TYPE 37, NOT USED HERE
         B     RECCOPY               * RECORD TYPE 38
         DC    7S(X'7F0'(4),RECEND) * RECORD TYPES 39 - 3F, NOT USED
* ABOVE INSTRUCTION IS EQUIVALENT TO 7 BRANCHES TO RECEND
         B     REC40                 * RECORD TYPE 40
         B     RECEND                * RECORD TYPE 41, NOT USED HERE
*-------------------------------------------------------------------*
*      RECORD PROCESSOR FOR 01, 03 RECORDS                          *
*      - COPY AT MOST 24 BYTES OF MESSAGE TEXT                      *
*-------------------------------------------------------------------*
REC0103  DS    0H                    * PROCESS RECORDS 01, 03
         LA    PBLDREC,RECAREA       * POINT TO OUTPUT BUFFER
         LH    R2,RECPRELL(,RECPTR) * LOAD RECORD PREFIX LENGTH
         LH    R7,RECLL(R2,RECPTR)  * SAVE LENGTH OF 1ST SEGMENT
         LH    R5,RECLL(,RECPTR)    * CALCULATE TEXT (REMAINING)
         SLR   R5,R2                 *     LENGTH
         LA    R8,24
         CR    R5,R8                 * MORE THAN 24 BYTES ?
         BNH   LESS24                * NO, USE THIS LENGTH
*-------------------------------------------------------------------*
*      - RECORD MUST BE TRUNCATED.                                  *
*        CALCULATE TEXT LENGTH OF ALL SEGMENTS                      *
*        TO INDICATE THAT RECORD WAS TRUNCATED, CHANGE SIGN OF LENGTH *
*-------------------------------------------------------------------*
         LA    R7,0(R2,RECPTR)       * POINT TO 1ST TEXT SEGMENT
         S     R5,TYPE04             * SUBTRACT SEQUENCE NUMBER FROM LEN
         SLR   R3,R3                 * ZERO TEXT LENGTH COUNTER
NEXTSEG  LH    R4,0(R3,R7)           * LOAD TEXT SEGMENT LENGTH
         LTR   R4,R4                 * TEST FOR ZERO
         BZ    ALLSEGS               * SKIP IF ZERO
         BXLE  R3,R4,NEXTSEG         * ACCUMULATE LENGTH AND BRANCH
         SLR   R3,R4                 * CORRECT VALUE AFTER BRANCH TEST
ALLSEGS  LNR   R7,R3                 * INDICATE RECORD WAS TRUNCATED
         LR    R5,R8                 * ONLY USE 1ST 24 BYTES OF TEXT
LESS24   ALR   R5,R2                 * GET WHOLE LENGTH TO MOVE
         LR    R0,R5                 * SAVE LENGTH
         LR    R2,PBLDREC            * POINT TO TARGET AREA
         LR    R4,RECPTR             * POINT TO SOURCE AREA
         LR    R3,R5                 * SET LENGTH OF SOURCE
         MVCL  R2,R4                 * MOVE RECORD
         STH   R0,RECLL(,PBLDREC)   * ADJUST TARGET LENGTH
         LH    R2,RECPRELL(,RECPTR) * LOAD RECORD PREFIX LENGTH
         STH   R7,RECLL(R2,PBLDREC) * ADJUST TARGET TEXT LENGTH
         B     RECEND                * END PROCESS RECORDS 01, 03
*-------------------------------------------------------------------*
*      RECORD PROCESSOR FOR 06, 07, 31, 34, 36, 38, AND 42 RECORDS    *
*      - COPY RECORD AS IT IS                                       *
*-------------------------------------------------------------------*
```

```
RECCOPY  DS     0H                   * RECORDS TO COPY
         LR    PBLDREC,RECPTR       * POINT TO INPUT RECORD
         B     RECEND               * END PROCESS COPY-ONLY RECORDS
*----------------------------------------------------------------------*
*      RECORD PROCESSOR FOR TYPE 40 (CHECKPOINT) RECORDS               *
*      - COPY SUBTYPES 01, 03, 04, 98                                  *
*      - IGNORE THE REST                                               *
*----------------------------------------------------------------------*
REC40    DS    0H                   * RECORD 40 - CHECKPOINT
         SR    R7,R7                * CLEAR WORK REGISTER
         IC    R7,RECSUBT(,RECPTR)  * GET RECORD SUBTYPE
         C     R7,TYPE03            * CNT TYPE RECORD ?
         BE    REC40USE             * YES, GO COPY IT
         C     R7,TYPE04            * SMB TYPE RECORD ?
         BE    REC40USE             * YES, GO COPY IT
         C     R7,TYPE01            * START CHECKPOINT RECORD ?
         BE    REC40USE             * YES, GO COPY IT
         C     R7,TYPE98            * END CHECKPOINT RECORD ?
         BNE   RECEND               * NO, IGNORE IT
REC40USE DS    0H                   * YES,
         LR    PBLDREC,RECPTR       *   INDICATE TO COPY RECORD
*----------------------------------------------------------------------*
*      CHECK IF ANYTHING INTERESTING FOUND                             *
*      IF SO, PUT THE RECORD                                           *
*----------------------------------------------------------------------*
RECEND   DS    0H                   * END PROCESS RECORDS
         LTR   PBLDREC,PBLDREC      * ANYTHING INTERESTING FOUND ?
         BZ    EPILOG               * NO, SKIP TO EPILOG
         LA    R1,DYNDCB            * YES, LOAD DCB ADDRESS
         USING IHADCB,R1            * LOCATE DCB
         CLC   RECLL(2,PBLDREC),DCBLRECL * IS DEFINED LRECL BIG ENOUGH?
         BH    SYNAD                * NO, TREAT AS I/O ERROR
         PUT   (1),(PBLDREC)        * YES, PUT RECORD
         DROP  R1                   * DROP BASE REG FOR DCB
         B     EPILOG               * SKIP TO EPILOG
*----------------------------------------------------------------------*
*  SYNAD EXIT - SEND A MSG, CLOSE, AND DEACTIVATE                      *
*----------------------------------------------------------------------*
SYNAD    WTO   'IMSE003 - ERROR DURING PUT TO IMSLOG',ROUTCDE=11,DESC=7
         L     ENTIND,TERMCALL      * INDICATE TO TERMINATE
         B     TERMCASE             * SKIP TO CLOSE AND TERMINATE
*----------------------------------------------------------------------*
*  END SYNAD EXIT                                                      *
*----------------------------------------------------------------------*
*----------------------------------------------------------------------*
*                                                                      *
*      TERMINATE CALL                                                  *
*                                                                      *
*----------------------------------------------------------------------*
TERMCASE C     ENTIND,TERMCALL      * IS THIS THE TERMINATE CASE ?
         BNE   OTHCASE              * IF NOT SKIP ON
         OI    DYNCLOSE,X'80'       * SET HIGH ORDER BIT IN CLOS LIST
         LA    R5,DYNDCB            * LOCATE DCB
         CLOSE ((R5)),MF=(E,DYNCLOSE) * CLOSE IT
         B     EPILOG               * END OF TERMINATE CASE
*----------------------------------------------------------------------*
*                                                                      *
*      OTHER CALL, ISSUE MESSAGE AND RETURN                            *
*                                                                      *
*----------------------------------------------------------------------*
OTHCASE  DS    0H                   * START OF OTHER CASE
```

```
*                                   * ERROR, SEND A MESSAGE
         WTO   'IMSE004 - INVALID CALL TYPE',ROUTCDE=11,DESC=7
         B     NOFREEMN             * SKIP TO TERMINATE
*---------------------------------------------------------------------*
*                                                                     *
*      EPILOG                                                         *
*      - FREEMAIN STORAGE FOR TERMINATE  CALL                         *
*                                                                     *
*---------------------------------------------------------------------*
EPILOG   DS    0H
         LR    R1,R13               * POINT TO DYNAMIC AREA
         L     R13,SAVEAREA+4       * POINT TO OLD SAVE AREA
         C     ENTIND,TERMCALL      * IS THIS TERMINATION CALL ?
         BNE   NOFREEMN             * NO, DON'T FREE STORAGE
         L     R0,SIZEWORK          * PICK UP LENGTH OF DYNAMIC AREA
         FREEMAIN R,LV=(0),A=(1)    * FREE IT
         SLR   R15,R15              * GET A ZERO
         ST    R15,0(,R11)          * STORE IT INTO THE USER WORD
NOFREEMN SLR   R15,R15              * CLEAR RETURN CODE
         L     R14,12(,R13)         * RESTORE RETURN REGISTER
         LM    R0,R12,20(R13)       * RESTORE OTHER REGS
         BR    R14                  * RETURN TO CALLER
*---------------------------------------------------------------------*
*                                                                     *
*      STATIC DATA AREA                                               *
*                                                                     *
*---------------------------------------------------------------------*
         DS    0F
INITCALL DC    F'1'
NORMCALL DC    F'2'
TERMCALL DC    F'3'
TYPE01   DC    XL4'01'
TYPE03   DC    XL4'03'
TYPE04   DC    XL4'04'
TYPE42   DC    XL4'42'
TYPE98   DC    XL4'98'
         DS    0F
SIZEWORK DC    AL1(0)
         DC    AL3(((ENDWORKA-WORKAREA+7)/8)*8)
         DS    0D
         PRINT NOGEN
LISTDCB  DCB   MACRF=PM,DDNAME=IMSLOG,DSORG=PS,EXLST=EXITLIST,        *
               SYNAD=SYNAD
LENDCB   EQU   *-LISTDCB             * LENGTH OF DCB
EXITLIST DC    XL1'85',AL3(DCBEXIT)  * DCB EXIT ADDRESS
         DCBD  DSORG=PS
*---------------------------------------------------------------------*
*                                                                     *
*      DCB EXIT                                                       *
*      - FORCE RECFM = VB                                             *
*      - ENSURE LRECL AND BLOCK SIZE ARE LARGE ENOUGH                 *
*                                                                     *
*---------------------------------------------------------------------*
IMSEXIT  CSECT
DCBEXIT  DS    0H
         USING *,R15                * SET ADDRESSABILITY
         LR    DCBPTR,R1            * LOAD DCB POINTER
         USING IHADCB,DCBPTR        * LOCATE DCB
         NI    DCBRECFM,DCBRECV+DCBRECSB+DCBRECBR
*                                   * SET NOT NEEDED FLAGS OFF
         OI    DCBRECFM,DCBRECV+DCBRECBR * SET RECFM=VB
         CLC   DCBBLKSI,IMSBLOCK    * IS BLOCK SIZE
         BNL   BLOCKOK              *   GREAT ENOUGH ?
         MVC   DCBBLKSI,IMSBLOCK    * NO, SET TO USUAL SIZE
BLOCKOK  EQU   *                    * YES, OK
         CLC   DCBLRECL,TESTLREC    * IS LRECL
         BNL   LRECLOK              *   GREAT ENOUGH ?
         MVC   DCBLRECL,MAXLRECL    * NO, SET TO MAX VALUE
LRECLOK  EQU   *                    * YES, OK
         LH    R9,DCBLRECL          * LOAD LRECL
         S     R9,BDWLEN            * SUBTRACT BDW LENGTH
         CH    R9,DCBBLKSI          * LRECL &gt; BLOCK SIZE - 4 ?
         BNH   SPANNOK              * NO, SKIP ON
         OI    DCBRECFM,DCBRECSB    * YES, FORCE SPANNED RECORDS
SPANNOK  EQU   *                    * SPANNED FLAG OK
         DROP  R15                  * DROP BASE REG
         BR    R14                  * RETURN TO OPEN

MAXLRECL DC    H'32756'
IMSBLOCK DC    H'6144'
TESTLREC DC    H'6140'
BDWLEN   DC    F'4'
*---------------------------------------------------------------------*
*  END DCB EXIT                                                       *
*---------------------------------------------------------------------*
*---------------------------------------------------------------------*
*                                                                     *
*      DYNAMIC WORK AREA                                              *
*                                                                     *
*---------------------------------------------------------------------*
WORKAREA DSECT
         DS    0F
SAVEAREA DS    18F
PARMLIST DS    3F
DYNDCB   DCB   MACRF=PM,DDNAME=IMSLOG,DSORG=PS,EXLST=EXITLIST
DYNOPEN  OPEN  (,),MF=L
DYNCLOSE CLOSE (,),MF=L
RECAREA  DS   0D
         DS   128CL256
ENDWORKA EQU   *
IMSEXIT  CSECT
R0       EQU   00                      EQUATES FOR REGISTERS 0-15
R1       EQU   01
R2       EQU   02
R3       EQU   03
R4       EQU   04
R5       EQU   05
R6       EQU   06
R7       EQU   07
R8       EQU   08
R9       EQU   09
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
DCBPTR   EQU   R2
RECPTR   EQU   R6
ENTIND   EQU   R10
PBLDREC  EQU   R9
*---------------------------------------------------------------------*
```

```
*                                                                      *
*      IMS RECORD MAPPING                                              *
*                                                                      *
*----------------------------------------------------------------------*
RECORD   EQU   0                    * START OF RECORD
RECLL    EQU   RECORD               * RECORD LENGTH
RECTYPE  EQU   RECORD+4             * RECORD TYPE
RECSUBT  EQU   RECORD+5             * RECORD SUBTYPE
RECPRELL EQU   RECORD+16            * TOTAL RECORD PREFIX LENGTH
         END   IMSEXIT
```

Chapter 20. Log Filter Exit Routine (DFSFTFX0)

The RSR Log Filter exit routine allows you to control the amount of log data sent to the tracking subsystem, by acting as a filter. Thus you can choose which log records to send to the tracking site. IMS supplies a default filter exit routine, which eliminates database records for:

- Databases not defined as covered
- Diagnostic data
- Block padding data

In this Chapter:

About This Routine

You can replace the IMS default filter exit routine with one of your own. Your replacement exit routine must return valid IMS log records, valid log record lengths, and valid IMS log record sequence numbers. The exit routine can only filter a particular log record by replacing it with a X'4304' log record containing the same log sequence number as the record being filtered.

Keep in mind that your replacement exit routine's performance can affect both RSR and the active IMS subsystem's logging.

The Log Filter exit routine is called in the following three cases:

- IMS and ILS Initialization

  The Log Filter exit routine is called during IMS initialization and during isolated log sender (ILS) instance initialization. You can use the log filter exit to perform setup or initialization work during IMS and ILS initialization.

  The initialization call can return a token, which is passed to the filtering exit routine on each subsequent call. The second word of the parameter list, upon return, contains a 0 or the address of the token.
- Log Buffer Send

  The Log Filter exit routine is called each time a log buffer is to be sent to the tracking site, so that you can filter which log records you want to be sent.
- IMS and ILS Termination

  The Log Filter exit routine is called during IMS termination and during isolated log sender (ILS) instance termination. You can use the log filter exit to perform cleanup or termination work during IMS and ILS termination.

The Log Filter exit routine should take into account the fact that it is possible to have multiple instances of the isolated log sender in one address space.

The mapping definition for the X'4304' log record is contained in the source code of the IMS DFSLOG43 macro. See "IMS-Supplied Filter Exit Routine" on page 176 for more information.

You must write this exit routine so that it is reentrant. It must be compiled with AMODE=31 and RMODE=ANY. It runs in supervisor state in user protect key 7. Its primary address space can be either CTL, DBCTL, DCCTL, batch, or isolated log sender.

The routine can be given process mode of SRB or TCB for log buffer send (see "Initialization and Termination Calls" on page 176 for the attributes of the routine for initialization and termination calls). Because an enabled unlock task (EUT) functional recovery routine (FRR) exists, no SVCs can be issued by the routine, nor can it hold any locks.

The input parameters are available until the exit routine returns to its caller; at that time, the storage is re-used. This means that the exit routine must copy any data it needs to preserve.

The log filter exit must ensure that filtered data has a format appropriate for the level of IMS that generated the data. X'4304' log records, for instance, must match the X'4304' log record DSECT for that level of IMS.

Use the RELEASE parameter of the ILOGREC macro to generate DSECTs for log records of different levels of IMS.

Restriction:IMS Callable Services are not applicable for use with this exit routine.

Communicating with IMS

IMS communicates with this exit through the entry and exit registers.

Contents of Registers at Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of a standard parameter list (a series of words containing the parameters' addresses)
13	MVS standard save area address (not an IMS pre-chained save area)
14	Return address (set with the assembly `BALR` instruction)
15	Filter exit routine entry address

The parameters are:

1. Function Code.

   This function code is one word in length and is set to the following:
   - X'0' for IMS or ILS initialization.
   - X'4' for log buffer send.
   - X'8' for IMS or ILS termination.
2. Optionally created token from initialization entry point call.

   If you do not provide a token, this parameter value is 0.
3. Log data being sent (from 0 to *N* log records).

   For both initialization and termination calls, this parameter value is 0.
4. One-word length of log data in bytes.

   For both initialization and termination calls, this parameter value is 0.

5. Data area.

   The exit routine moves the log data to this area. This area must be large enough to contain the unfiltered log data. For both initialization and termination calls, this parameter value is 0.
6. Sink area data length in words.

   This is filled in by the exit routine. For both initialization and termination calls, this parameter value is 0.
7. Package descriptor.

   The package descriptor contains information about the log data. If the filter exit will encounter data from more than one level of IMS, reference the package descriptor to determine which level of IMS generated the log data.

   Two formats of package descriptor are possible:
   - Log data from IMS is accompanied by a package descriptor mapped by the DFSPKR macro.
   - Data from ILS is accompanied by a package descriptor described by the IPD DSECT in the DFSILT macro.

   The halfword at offset 2 will contain a 1 for the DFSPKR type and a 2 for the IPD type.

   The level of IMS that generated the log data is indicated by the value in field PKRRELSE (DFSPKR) or IPDRELSE (DFSILT). The values correspond to values defined in the DFSLOGRC macro.

   Whether or not log data from more than one level of IMS will be encountered, the filter exit must ensure that filtered data has a format appropriate for the level of IMS that generated the data.

   Use the RELEASE parameter of the ILOGREC macro to generate DSECTs for log records of different levels of IMS.

Contents of Registers at Exit

Before returning to IMS, the exit routine must restore all registers except for registers 0, 1, and 15.

Also before returning to IMS, be sure to set the token in the parameter list (on the initialization call) if you want to use a token, and be sure the data and data length have been set (on the send call). See “Contents of Registers at Entry” on page 174 for more on the parameter list.

Recovery Environment

For initialization and termination calls, an ESTAE is established for the recovery environment; for log buffer send calls, a functional recovery routine (FRR) is established for the recovery environment. If the FRR exit routine is driven, it will SDUMP. If retry is allowed, it turns off filtering for the duration of the IMS instance. (This exit routine is not reloaded as a result of a `/START SERVGRP` or `/STOP SERVGRP` command.) If an error is detected in the filtered record, filtering is turned off for the duration of the IMS instance. If retry is not allowed, the component abends.

Initialization and Termination Calls

IMS calls the Log Filter exit routine during IMS initialization and normal termination and for ILS instance initialization and termination. You can thereby prepare for log filtering or clean up after termination. If you do not return a token on the initialization call, IMS sets the token to zero for all subsequent filter exit routine calls.

The attributes of the routine for initialization and termination is the same as for log buffer send, except that the routine can only be given a process mode of TCB, thus allowing SVC calls.

The recovery environment for initialization and termination calls is different than for log buffer send calls. An ESTAE environment is created to cover the initialization and termination call. If the ESTAE is driven and retry is allowed, it turns off log filtering for the duration of the IMS instance and issues an error message. If retry is not allowed, the component abends (under certain circumstances, such as CANCEL, MVS does not allow a retry).

IMS-Supplied Filter Exit Routine

This exit routine includes a summary of which log records can be filtered by the Log Filter exit routine. It also summarizes which log records can be filtered if you do not need to restart data communications at the tracking site.

The filter exit module supplied with IMS is table driven, with the tables already set up to filter some log records. These records are replaced with a X'4304' dummy log record. The log filter exit module contains complete information about which log records can be eliminated or changed. The name of this module is DFSFTFX0. Use the prolog and tables in the exit routine as a reference.

The module also indicates how to eliminate data communication log records. This causes filtering of message queue records, scratch pad areas, fast path output messages, DC sequence number records, and DC-related checkpoint records. Using this option reduces log volume but requires a COLDCOMM emergency restart at the tracking site after remote takeover to restart data communications.

Recommendation: You should be very careful when writing the replacement for the Log Filter exit routine, because incorrect filtering of the log data can make the tracking SLDS data invalid or unusable. You also need to consider that multiple copies of the exit routine can run for concurrent IMS jobs and ILS instances.

Using the Sample Log Filter Exit Routine

For the latest version of DFSFTFX0, see the IMS.SVSOURCE library; member name is DFSFTFX0.

Chapter 21. Logger Exit Routine (DFSFLGX0)

You can use the Logger exit routine for recovery purposes.

In this Chapter:

About This Routine

You can write a Logger exit routine that is called during IMS logger execution. IMS passes the exit routine all log data after the data has been written to the IMS log. Your exit routine can then process this data for recovery purposes.

IMS calls the Logger exit routine with an *initialization call* when the logger is opened and with a *termination call* when the logger is closed. At these times, your exit routine can get or free any additional storage that it needs to run. IMS also calls the exit routine and passes log data to it with a *write call* whenever a block of data is written to the logger.

This exit routine is optional. No default and no sample are provided. Be aware that the interface to this exit routine may change in future releases of IMS.

Table 26 shows the attributes of the Logger exit routine.

Table 26. Logger Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSFLGX0.

Link editing	Follow the guidelines described in "Link-Editing the Routines" on page 5.
	
	You must link-edit the exit routine into IMS.RESLIB (or a library concatenated with it) as a separate **reentrant** load module. If the module is not present in the load library, the IMS logger does not load or call it.
	
	Example: This shows you how to link-edit the exit routine into IMS.RESLIB.
	`//LINKIT   JOB  1,MSGLEVEL=1`
	`//LINK     EXEC PGM=IEWL,PARM=RENT`
	`//SYSUT1   DD   UNIT=SYSDA,SPACE=(TRK,(20,20))`
	`//SYSPRINT DD   SYSOUT=A`
	`//SYSLMOD  DD   DSN=IMS.RESLIB.,DISP=SHR`
	`//OBJIN    DD   DSN=IMS.USERLIB.,DISP=SHR`
	`//SYSLIN   DD   *`
	`    INCLUDE OBJIN(DFSFLGX0)`
	`    MODE    AMODE(31),RMODE(ANY)`
	`    NAME    DFSFLGX0(R)`
	`/*`

Including the routine	No special steps are needed to include this routine.
IMS callable services	This exit routine can use IMS Callable Storage Services. This exit routine is defined to IMS as an IMS standard user exit. Exit routines that are defined to IMS receive the callable services token in the standard exit parameter list. This exit routine does not need to issue an initialization call (DFSCSII0) to use IMS Callable Services. This exit routine must be manually link-edited with DFSCSI00. For guideline on using IMS Callable Services, see "Using IMS Callable Services" on page 8.

Table 26. Logger Exit Routine Attributes (continued)

Attribute	Description
Sample routine location	No sample exit routine is provided.

Attributes of the Routine

The Logger exit routine must be written as **reentrant**. The exit routine receives control running in 31-bit addressing mode and must return control in that mode. It is called in TASK mode, with no locks held, and in non-cross memory, non-AR mode. In an online IMS environment, the Logger exit routine runs in key 7, supervisor state, in the IMS control region address space. In batch and log recovery environments, it runs in key 8, problem state.

This information on various IMS environments is for the current release of IMS and might change in subsequent releases.

Calling the Routine

The Logger exit routine is called at each of the following times. When IMS calls the routine determines the “call type.”

Initialization Call

IMS calls DFSFLGX0 when the logger is initialized. IMS makes this call when it opens the first output log.

An initialization call (call type 1) is made for the following:

- Normal initialization (DB/DC, DBCTL, DCCTL, batch)
- Initialization of log recovery during emergency restart processing if log recovery from the write ahead data set (WADS) is required
- Log Recovery utility initialization (CLS mode)
- Alternate IMS system logger initialization when the alternate opens its first OLDS for output during an XRF takeover

OLDS/SLDS Write Call

IMS calls DFSFLGX0 after a block of data is successfully written to the online log data set (OLDS) or the system log data set (SLDS). The OLDS is accessed in a DB/DC, DBCTL, or a DCCTL environment and in the SLDS in a batch environment.

A pointer to the data that was written is passed to the exit routine. (The blocks of data might not be presented in sequence.) All processing of the data must be completed before returning to IMS, because the data address is not valid after leaving DFSFLGX0.

A write call (call type 2) is made for the following:

- Normal block write (a block of data is written to the log during normal IMS processing).
- Buffer purge (the final block(s) of log data are written to the log during IMS abnormal termination).
- Log recovery during emergency restart processing (blocks of data are recovered from the WADS and written to the OLDS).
- Log Recovery utility processing when recovering entries of complete log buffers from the WADS (CLS mode).

Under some abend or error conditions, one or more blocks might be written to the log and not passed to the exit routine. Under some abend or error conditions, IMS might pass the same blocks to DFSFLGX0 several times. Your exit routine must be able to tolerate both of these situations.

Termination Call

IMS calls DFSFLGX0 when the logger is terminated. IMS makes this call after it closes the output log and notifies DBRC.

A termination call (call type 3) is made for the following:

- Normal termination
- Abnormal termination (if a terminal call is possible)
- Termination of log recovery during emergency restart processing
- Log Recovery utility termination (CLS mode)

If IMS terminates abnormally, there might be cases when the logger is unable to make the termination call to DFSFLGX0. Therefore, your exit routine must be able to tolerate not being called for termination.

Restrictions

The Logger exit routine is subject to the following restrictions:

- This exit routine must not modify the log data passed to it on an OLDS/SLDS write call. It must not try to locate, access, or modify any IMS control blocks not specifically passed to it by IMS.
- All addresses, entry register contents, and parameter list contents can change from one call of DFSFLGX0 to the next call. One call should not depend on addresses from a prior call. Similarly, this exit routine must not assume what TCB it is running under, nor that the TCB is the same from one call to the next call.
- This exit routine can call only those MVS services that it is authorized to call. It must **not** call any internal IMS services.

Keep in mind that the IMS logger is critical to performance. Avoid coding the exit routine to do things that could negatively affect performance in the IMS logger, such as WAITs and other MVS services that could cause long delays before returning to your exit routine.

Communicating with IMS

IMS communicates with this routine through the entry registers, a parameter list, and the exit registers.

Content of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Content
1	Address of the standard exit parameter list.
13	Address of the save area. Your exit routine must not change the first three words of this save area. This save area is **not** chained to any other IMS save area.
14	Return address to IMS.

Register	Content
15	Entry point of this exit routine.

Standard Exit Parameter List

Table 27 shows the content of the standard exit parameter list. When the user exit is activated, IMS passes it the address of this list in register 1.

Table 27. Standard Exit Parameter List (Mapped by DFSSXPL)

Field	Offset	Length	Description
SXPLVER	0	4	Address of word containing version number of standard IMS user exit parameter list.
SXPLATOK	4	4	Address of word containing callable services token. The callable services token must be included in the callable services parameter list when a callable service is requested.
SXPLAWRK	8	4	Static address of 512-byte work area for this exit routine. This is a permanent storage area allocated during IMS initialization and passed to this exit routine each time it is called.
SXPLFSPL	12	4	Address of function-specific parameter list. The figures that follow show this list.
SXPLINTX	16	4	Address of user data tabled loaded by DFSINTX0 at IMS initialization time. This field is valid in only IMS environments where DFSINTX0 is called. It will be zero in any other environment. For information about the DFSINTX0 exit routine, see “Chapter 36. Initialization Exit Routine (DFSINTX0)” on page 289.

Function-Specific Parameter List on Entry

The content of the function-specific parameter list depends on whether this exit routine is called by a call type 1, 2, or 3. Table 28 through Table 30 outline the contents of the parameter list for each of these calls.

Table 28. Function-Specific Parameter List for Initialization Call, Call Type 1 (Mapped by LGWXPLST, Which Is Included in LCDSECT)

Field	Offset	Length	Content
LGWXTYPE	0	1	Call type: 1
LGWXENVR	1	1	Environment type: **X'01'=** DB/DC online system **X'02'=** Batch IMS system (includes CICS/DLI) **X'03'=** Log Recovery utility **X'04'=** DBCTL system **X'05'=** DCCTL system

Table 28. Function-Specific Parameter List for Initialization Call, Call Type 1 (Mapped by LGWXPLST, Which Is Included in LCDSECT) (continued)

Field	Offset	Length	Content
LGWXFLG1	2	1	Flag byte: **X'10'** 0 = not XRF takeover, 1 = XRF takeover **X'20'** 0 = not /ERE log recovery, 1 = /ERE log recovery **X'08'** 1 = field LGWXTODN exists All other flag bits reserved.
	3	1	Reserved.
LGWXTOD	4	8	This field has been left here for backwards compatibility. The old timestamp format value is in the 00YYDDDF HHMMSSTF format.
LGWXDODN	1C	12	This field contains the current date and time fields, but in the IMS internal packed-decimal format. For further information on the internal packed-decimal time-stamp format, see *IMS/ESA Application Programming: Transaction Manager* or *IMS/ESA DBRC Guide and Reference*.
LGWXSSID	C	8	IMS subsystem ID (IMSID for non-XRF, RSE name for XRF).
LGWXBUFR	14	4	Unused on this call.
LGWXBSIZ	18	4	Unused on this call.
LGWXTMST	1C	12	This field contains the current date and time fields, but in the IMS internal packed-decimal format. For further information on the internal packed-decimal time-stamp format, see *IMS/ESA Application Programming: Transaction Manager* or *IMS/ESA DBRC Guide and Reference*. See the LGWXTOD field above for the timestamp format used prior to IMS/ESA V6.

Table 29 shows the parameter list for call type 2.

Table 29. Function-Specific Parameter List for OLDS/SLDS Write Call, Call Type 2 (Mapped by LGWXPLST, Which Is Included in LCDSEC

Field	Offset	Length	Content
LGWXTYPE	0	1	Call type: 2
LGWXENVR	1	1	Environment type: **X'01'=** DB/DC online system **X'02'=** Batch IMS system (includes CICS/DLI) **X'03'=** Log Recovery utility **X'04'=** DBCTL system **X'05'=** DCCTL system

Table 29. Function-Specific Parameter List for OLDS/SLDS Write Call, Call Type 2 (Mapped by LGWXPLST, Which Is Included in LCDSEC (continued)

Field	Offset	Length	Content
LGWXFLG1	2	1	Flag byte: **X'20' 0 =** Not /ERE log recovery, 1 = /ERE log recovery **X'08'** 1 = field LGWTMST exists
	3	1	Reserved
LGWXTOD	4	8	This field has been left here for backwards compatibility. The old timestamp format value is in the 00YYDDDF HHMMSSTF format.
LGWXTODN	1C	12	This field contains the current date and time fields, but in the IMS internal packed-decimal format. For further information on the internal packed-decimal time-stamp format, see *IMS/ESA Application Programming: Transaction Manager* or *IMS/ESA DBRC Guide and Reference*.
LGWXSSID	C	8	IMS subsystem ID.
LGWXBUFR	14	4	Address of IMS log block data that has been successfully written to the OLDS/SLDS. (This might be a copy of the original IMS buffer.)
LGWXBSIZ	18	4	Length of log data, in bytes.
LGWXTMST	1C	12	This field contains the current date and time fields, but in the IMS internal packed-decimal format. For further information on the internal packed-decimal time-stamp format, see *IMS/ESA Application Programming: Transaction Manager* or *IMS/ESA DBRC Guide and Reference*. See the LGWXTOD field above for the timestamp format used prior to IMS/ESA V6.

Table 30 shows the parameter list for call type 3.

Table 30. Function-Specific Parameter List for Termination Call, Call Type 3 (Mapped by LGWXPLST, Which Is Included in LCDSECT)

Field	Offset	Length	Content
LGWXTYPE	0	1	Call type: 3
LGWXENVR	1	1	Environment type: **X'01'=** DB/DC online system **X'02'=** Batch IMS system (includes CICS/DLI) **X'03'=** Log Recovery utility **X'04'=** DBCTL system **X'05'=** DCCTL system

Table 30. Function-Specific Parameter List for Termination Call, Call Type 3 (Mapped by LGWXPLST, Which Is Included in LCDSECT) (continued)

Field	Offset	Length	Content
LGWXFLG1	2	1	Flag byte: **X'80'** 0 = normal termination, 1 = abnormal termination **X'40'** 0 = buffer purge succeeded, 1 = buffer purge failed, (abend only) **X'20'** 0 = not /ERE log recovery, 1 = /ERE log recovery **X'08'** 1 = field LGWTODN exists All other flag bits reserved.
	3	1	Reserved.
LGWXTOD	4	8	This field has been left here for backwards compatibility. The old timestamp format value is in the 00YYDDDF HHMMSSTF format.
LGWXTODN	1C	12	This field contains the current date and time fields, but in the IMS internal packed-decimal format. For further information on the internal packed-decimal time-stamp format, see *IMS/ESA Application Programming: Transaction Manager* or *IMS/ESA DBRC Guide and Reference*.
LGWXSSID	C	8	IMS subsystem ID.
LGWXBUFR	14	4	Unused on this call.
LGWXBSIZ	18	4	Unused on this call.
LGWXTMST	1C	12	This field contains the current date and time fields, but in the IMS internal packed-decimal format. For further information on the internal packed-decimal time-stamp format, see *IMS/ESA Application Programming: Transaction Manager* or *IMS/ESA DBRC Guide and Reference*. See the LGWXTOD field above for the timestamp format used prior to IMS/ESA V6.

For calls made during normal IMS operation, the time in the field at offset X'10' contains the start time of the **current** IMS system. For calls made during emergency restart log recovery, this field contains the start time of the **previous** IMS system, the one whose log is being recovered.

If log recovery is required during emergency restart processing, DFSFLGX0 is called for two sets of initialization/write/termination call sequences. The first set of calls occurs during log recovery and sets a flag indicating that log recovery is processing. Only the data from the buffers (recovered from the WADS and written to the OLDS to close it) is passed. The second set of calls occurs for normal IMS processing.

Content of Registers on Exit

Before returning to IMS, the exit routine must restore all registers except for register 15, which must contain the following:

Register	Contents
15	0

Chapter 22. Non-Discardable Messages Exit Routine (DFSNDMX0)

The Non-Discardable Messages exit routine provides users with a mechanism to tell IMS what to do with the input message associated with an abended application program. Otherwise, IMS can arbitrarily discard messages from the system and issue message DFS555I.

In this Chapter:

About This Routine

The Non-Discardable Messages exit routine receives control when an IMS application abends with an input message in process.

Table 31 shows the attributes of the Non-Discardable Messages exit routine.

Table 31. Non-Discardable Messages Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSNDMX0.
Link editing	Follow the guidelines described in "Link-Editing the Routines" on page 5. This exit routine must be reentrant. It executes in non-cross-memory mode.

Including the routine	If you write your own exit routine and plan to use IMS callable services, you must manually link edit the routine with DFSCSI00 and you must link the routine with IMS.RESLIB. The following example shows the required link-edit JCL statements.
	`INCLUDE LOAD(DFSNDMX0)`
	`INCLUDE LOAD(DFSCSI00)`
	`ENTRY   DFSNDMX0`
	`NAME    DFSNDMX0(R)`

IMS callable services	To use callable services with this routine examine the value of the SXPLATOK field in the IMS standard user exit parameter list to see if a callable services token is passed to the routine. • If SXPLATOK is zero, you cannot use callable services with this routine. • If SXPLATOK is non-zero, the callable services token is included, and you can use callable services. You can use the 256-byte work area addressed by SXPLAWRK in the standard user exit parameter list to call DFSCSIF0. See "IMS Standard User Exit Parameter List" on page 4 for more information on the standard user exit parameter list. See "Using IMS Callable Services" on page 8 for more information in IMS callable services.
Sample routine location	IMS.TMSOURCE (member name DFSNDMX0). The mapping of the NDM interface block is available from the IMS library IMS.GENLIBB (member name DFSNDM).

Restrictions

Not all destinations are valid alternates for input messages. If you use this exit routine to requeue messages to alternate destinations, see "Queue the Message to an Alternate Destination" on page 186 for information on valid destinations.

Processing Options

The following sections describe the valid processing options for DFSNDMX0. If you request an option that is not valid, IMS ignores your request and continues normal processing (the default option).

Continue normal processing

Continue normal processing is the default option. Request this option by setting register 15 to zero before returning to IMS. IMS proceeds as if this exit routine had not been called.

Depending on the type of application abend that initiated the exit routine, IMS might delete the message, issue a DFS555I message to the originating terminal and master terminal, and issue a DFS554A message to the master terminal.

Delete the Input Message from the System

Request this option by setting register 15 to 4 before returning to IMS. If you request this option, IMS does the following:

1. Issues a DFS555I message to the originating terminal (if possible) and to the master terminal
2. Deletes the input message from the system
3. Issues a DFS554A message to the master terminal

Queue the Message to the Suspend Queue

Request this option by setting register 15 to 8 before returning to IMS. If you request this option, IMS queues the input message to the suspend queue of the transaction that was being processed when the application abended. IMS suspends the transaction and, depending on the type of abend, might issue a DFS554A message to the master terminal.

Requeue the Input Message to the Original Transaction

Request this option by setting register 15 to 12 before returning to IMS. If you request this option, IMS queues the input message to the normal processing queue of the transaction that was being processed when the application abended. IMS USTOPs the transaction unless directed to do otherwise by the contents of NDMTRNST and, depending upon the type of abend, might issue a DFS554A message to the master terminal.

Queue the Message to an Alternate Destination

Request this option by setting register 15 to 16 before returning to IMS and placing a valid destination name in the NDMDEST field of the NDM interface block. Table 32 shows the valid destination types and how to specify them in NDMDEST.

Table 32. Valid Alternate Destinations

Alternate Destination	NDMDEST Value
LTERM	Specify a local, remote, or ETO LTERM, using the LTERM name or ETO user descriptor name.
OTMA	Specify the OTMA TPIPE name, or a name that is meaningful to the OTMA exit routines.
LU 6.2	Specify a local LU 6.2 device descriptor. The LU 6.2 device must be on the local IMS subsystem.

Table 32. Valid Alternate Destinations (continued)

Alternate Destination	NDMDEST Value
Transaction	Specify a local or remote transaction code. The following transaction types are **not** valid destinations: • Fast Path exclusive transaction. • Conversational transaction. • SAA communications-driven transaction (that is, a CPI-C driven transaction). If you specify an invalid transaction type, IMS ignores the request and continues normal processing.

If NDMDEST contains an invalid destination, such as zeros or blanks, IMS ignores the request to change the destination and continues normal processing.

If NDMDEST contains a destination that is unknown to IMS, processing depends on whether OTMA, and ETO or shared queues are active.

With OTMA, and ETO or shared queues active
IMS invokes the OTMA exit routines before invoking the Output Creation exit routine (DFSINSX0).

Without OTMA, ETO, or shared queues
IMS ignores the request and continues normal processing.

When IMS requeues the input message to a valid destination, IMS completes the message processing as follows:

1. Issues a DFS550I message (succeeded version) to the master terminal
2. Issues a DFS555I message to the originating terminal (if possible) and to the master terminal
3. Deletes the input message from the abended transaction
4. Issues a DFS554A message to the master terminal

Related Reading:

- For more information on OTMA exit routines—

  "Chapter 48. OTMA Destination Resolution Exit Routine (DFSYDRU0)" on page 349

  "Chapter 49. OTMA Input/Output Edit Exit Routine (DFSYIOE0)" on page 353

  "Chapter 50. OTMA Prerouting Exit Routine (DFSYPRX0)" on page 357
  IMS/ESA Open Transaction Manager Access Guide
- For more information on DFSINSX0 and ETO—

  "Chapter 51. Output Creation Exit Routine (DFSINSX0)" on page 361

  IMS/ESA Administration Guide: Transaction Manager
- For more information on shared queues—

  IMS/ESA Administration Guide: Transaction Manager

Communicating with IMS

This exit routine uses a parameter list, entry and exit registers, and the Non-Discardable Messages interface block (NDM) to communicate with IMS.

Contents of Registers on Entry

At entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Content
1	Address of the standard exit parameter list. See Table 33 for the contents of this parameter list.
13	Address of a single standard MVS save area.
14	Return address to IMS.
15	Entry point of this exit routine.

Standard User Exit Parameter List

Table 33 shows the contents of the standard exit parameter list. When the exit routine is invoked, IMS passes the address of this list to the exit routine in register 1.

Table 33. Standard Exit Parameter List (Mapped by DFSSXPL)

Field	Offset	Length	Description
SXPLVER	0	4	Address of word containing version number of the standard user exit parameter list.
SXPLATOK	4	4	Zero or the address of a word containing the callable services token, which must be included in the callable services parameter list when a callable service is requested.
SXPLAWRK	8	4	Address of a 512-byte work area for this exit routine.
SXPLFSPL	12	4	Address of the NDM interface block. See Table 34 on page 189 for the contents of this interface block.
SXPLINTX	16	4	Address of user data table loaded by DFSINTX0 at IMS initialization time. This field is valid only in IMS environments where DFSINTX0 is called. It will be X'80000000' in any other environment. For more information about the DFSINTX0 exit routine, see “Chapter 36. Initialization Exit Routine (DFSINTX0)” on page 289.

NDM Interface Block

Table 34 on page 189 shows the contents of the NDM interface block. The address of this parameter list is in the standard user exit parameter list (field name SXPLFSPL). The mapping of the NDM interface block is available from the IMS library IMS.GENLIBB (member name DFSNDM).

Related Reading:

- See *OS/390 V1R1 MVS Planning: APPC Management* for more information on the LU 6.2 device description fields in the NDM interface block.
- See *IMS/ESA Open Transaction Manager Access Guide* for more information on the OTMA description fields in the NDM interface block.

Table 34. NDM Interface Block

Field	Offset	Length	Content
NDMEYE	X'00'	4	NDM eye catcher
NDMTRAN	X'04'	8	Transaction that the application was processing when it abended. This transaction is associated with the input message pointed to by NDMMSGA.
NDMPSBN	X'0C'	8	PSB associated with the application that abended
NDMUSID	X'14'	8	User ID
NDMGRPNM	X'1C'	8	Group name
	X'24'	1	Reserved
NDMSRCFL	X'25'	1	A flag that indicates the origin of the input message. This flag is set with one of the following values: **Value** **Description** **0** NDMLTERM The source of the input message is an LTERM. Subsequent fields contain information about the LTERM. **1** NDMOTMA The source of the input message is OTMA. Subsequent fields contain information about the OTMA source. **2** NDMLU62 The source of the input message is an LU 6.2 device. Subsequent fields contain information about the LU 6.2 device.
NDMSRCIN	X'26'	1	Start of source description
NDMLTERM	X'26'	8	Name of the originating LTERM if NDMSRCFL is set to NDMLTERM (value 0)
NDMTPIPE	X'26'	8	OTMA TPIPE name if NDMSRCFL is set to NDMOTMA (value 1)
NDMMEM	X'2E'	16	OTMA member name
NDMTPSYN	X'3E'	1	OTMA TPIPE synchronization level
NDMMGSYN	X'3F'	1	OTMA message synchronization level
NDMLUNM	X'26'	8	LU name if NDMSRCFL is set to NDMLU62 (value 2)
NDMNWID	X'2E'	8	Network identifier
NDMSIDE	X'36'	8	APPC/MVS side information name
NDMMODE	X'3E'	8	VTAM mode table name
NDMTPNML	X'46'	2	Length of TP name contained in NDMTPNM
NDMTPNM	X'48'	64	TP name
NDMCONV	X'88'	1	APPC/MVS conversation type
NDMSYNC	X'89'	1	APPC/MVS synchronization level

Table 34. NDM Interface Block (continued)

Field	Offset	Length	Content
	X'8A'	18	Reserved
NDMABEND	X'9C'	4	Abend code in system format `00sssuuu`, where: **sss** MVS system abend code; **uuu** IMS user abend code
NDMTSLCL	X'A0'	8	The local time stamp of the arrival of the input message in the system. NDMTSLCL contains the two fields NDMDLCL and NDMTLCL.
NDMDLCL	X'A0'	4	The local date that the message arrived in the system. The date format is `YYYYDDDf`, where: **YYYY** Year; **DDD** Julian day; **f** X'F'
NDMTLCL	X'A4'	4	The local time that the message arrived in the system. The time format is `HHMMSSTf`, where: **HH** Hour; **MM** Minutes; **SS** Seconds; **T** Tenths of the second; **f** X'F'

Table 34. NDM Interface Block (continued)

Field	Offset	Length	Content
NDMTSUTC	X'A8'	12	The UTC time stamp of the arrival of the input message in the system. The time-stamp format is the following: **Year/day** `YYYYDDDf` **Time** `HHMMSSTHmiju` **Offset** `Aqq$` The time-stamp fields include the following: **YYYY** Year **DDD** Julian day **f** X'F' **HH** Hour **MM** Minutes **SS** Seconds **T** Tenths of the second **H** Hundredths of the second **m** Milliseconds **i** Tenths of a millisecond **j** Hundredths of a millisecond **u** Microseconds **A** Attribute of the time value **qq** Quarter-hours of offset from UTC **$** Decimal sign for the offset, either positive (X'C') or negative (X'D').
NDMSPAA	X'B4'	4	Address of the SPA if the transaction in NDMTRAN is a conversational transaction. Otherwise, this field contains zeros. If the SPA is present, the format is as follows: `LL ZZZZ transaction_code data` **LL** Two-byte length field that includes the length of LLZZZZ **ZZZZ** Four-byte field that always contains zeros **transaction_code** Eight-byte field that contains the transaction code for the conversation or blanks **data** SPA user data

Table 34. NDM Interface Block (continued)

Field	Offset	Length	Content
NDMMSGA	X'B8'	4	Address of the input message. The message format is as follows: `LL ZZ message-segment` **LL** Two-byte length field that includes the length of LLZZ **ZZ** Two-byte field that always contains zeros, except for the last message segment which contains X'FFFF' **message-segment input** message segment For a single-segment message: LL=NDMMSGL and ZZ=X'FFFF' For a multi-segment message, the pattern is: • NDMMSGA=address of first segment • NDMMSGA+LL=address of second segment • NDMMSGA+LL+LL= address of third segment
NDMMSGL	X'BC'	4	Total length of input message
	X'C0'	24	Reserved
NDMTRNST	X'D8'	4	Transaction status flag. DFSNDMX0 may set this field to any one of the following values. IMS examines this field upon return from DFSNDMX0. **Value** **Description** Do not (U)STOP the abended transaction and do not STOP the abended program. **2** Do not send the DFS555I message. **3** Do not (U)STOP the abended transaction and do not STOP the abended program, *and*, do not send the DFS555I message
NDMDEST	X'DC'	8	Name of the alternate destination to which the input message is to be queued. IMS only examines this field if you pass return code 16 in register 15. Otherwise, IMS ignores this field. See “Queue the Message to an Alternate Destination” on page 186 for more information on NDMDEST. See “Contents of Registers on Exit” for the valid return codes for register 15.

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers except register 15, which must contain one of the following return codes:

Return Code	Meaning
0	Continue normal processing.
4	Delete the input message from the system.
8	Queue the input message to the suspend queue.
12	Requeue the input message to the original transaction.
16	Queue the message to an alternate destination that is named in the NDMDEST field in the NDM interface block.

Chapter 23. Partner Product Exit Routine (DFSPPUE0)

This chapter documents General-Use Programming Interface and Associated Guidance Information.

Related Reading: See "Notices" on page xxi to understand this classification of information.

This is a user exit routine.

In this Chapter:

About This Routine

The Partner Product exit routine is entered immediately before IMS is ready for startup (before the DFS994I start complete message is issued). The exit routine is provided to allow the initialization of products that run with IMS. The exit routine can load or link one or more partner product routines. The exit routine is deleted after control returns to IMS.

Be aware that the interface to this exit routine may change in future releases of IMS.

The exit routine must reside on the library pointed to by the STEPLIB DD statement. If the exit routine exists, it is called.

Table 35 shows the attributes of the Partner Product exit routine.

Table 35. Partner Product Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSPPUE0.
Link editing	Follow the guidelines described in "Link-Editing the Routines" on page 5.
Including the routine	No special steps are needed to include this routine.
IMS callable services	This exit routine can use IMS Callable Storage Services. This exit routine is defined to IMS as an IMS standard user exit. Exit routines that are defined to IMS receive the callable services token in the standard exit parameter list. This exit routine does not need to issue an initialization call (DFSCSII0) to use IMS Callable Services. This exit routine must be manually link-edited with DFSCSI00. **Related Reading:**For guidelines on using IMS Callable Services, see "Using IMS Callable Services" on page 8.
Sample routine location	No sample exit routine is provided.

Communicating with IMS

IMS uses the entry registers, a parameter list, and the exit registers to communicate with the exit routine.

Content of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of the standard exit parameter list.
13	Address of the save area. Your exit routine must not change the first three words of this save area. This save area is **not** chained to any other IMS save area.
14	Return address to IMS.
15	Entry point of this exit routine.

Standard IMS User Exit Parameter List

Table 36 shows the content of the standard exit parameter list. When the user exit is invoked, IMS passes the address of this list to the exit routine in register 1.

Table 36. Standard Exit Parameter List (Mapped by DFSSXPL)

Field	Offset	Length	Description
SXPLVER	0	4	Address of word containing version number of standard IMS user exit parameter list.
SXPLATOK	4	4	Address of word containing callable services token, which must be included in the callable services parameter list when a callable service is requested.
SXPLAWRK	8	4	Static address of 512-byte work area for this exit routine. This is a permanent storage area allocated during IMS initialization and passed to this exit routine each time it is called.
SXPLFSPL	12	4	Address of function-specific parameter list (Table 37 shows this list).
SXPLINTX	16	4	Address of user data table loaded by DFSINTX0 at IMS initialization time. This field is valid in only IMS environments where DFSINTX0 is called. It will be zero in any other environment. For more information about the DFSINTX0 exit routine, see “Chapter 36. Initialization Exit Routine (DFSINTX0)” on page 289.

Function-Specific Parameter List on Entry

Table 37 shows the content of the function-specific parameter list. The address of this parameter list is in the standard IMS user exit parameter list.

Table 37. Function-Specific Parameter List for Partner Product Exit (Mapped by DFSPPUE)

Field	Offset	Length	Content
PPUEIMSD	0	4	IMS identifier
PPUEREL	4	1	IMS level
PPUETYP	5	1	IMS subsystem type
PPUEOSL	6	1	MVS level
Reserved	7	1	

Content of Registers on Exit

Before returning to IMS, the exit routine must restore all registers except register 15, which must contain one of the following return codes:

Return Code	**Meaning**
0	Processing continues.
non-0	IMS abends with U0740. The exit routine should return this code if critical tasks do not complete successfully.

Chapter 24. RECON I/O Exit Routine (DSPCEXT0)

The RECON I/O exit routine writes a journal that keeps track of changes to the RECON data set.

In this Chapter:

About This Routine

Using this exit routine enables you to keep track of changes to the RECON data set in the form of a journal. You can code this routine so that it updates the journal each time a record of the data set is updated, inserted, deleted, or read. You can also record changes that are internal to the RECON access modules, such as header record extension control item changes, or the addition and deletion of multiple update control records within the data set.

You can use the journal, in turn, as a trace facility, to monitor the activity of specific record types, or as a means of writing your own recovery utility for the RECON data set.

Related Reading: For additional detailed information about how DBRC accesses records on the RECON data set, see the *IMS/ESA Diagnosis Guide and Reference*.

Table 38 shows the attributes of the RECON I/O exit routine.

Table 38. RECON I/O Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DSPCEXT0.
Link editing	Follow the guidelines described in "Link-Editing the Routines" on page 5. You must write and link-edit this routine as **reentrant** (RENT). After assembling the source code, you need to link-edit the object code for this module into the IMS load module DSPCINT0.
Including the routine	No special steps are needed to include this routine.
IMS callable services	This exit is not eligible to use IMS Callable Services.
Sample routine location	IMS.SVSOURCE (member name DSPCEXT1)

Attributes of the Routine

You must write and link-edit this routine as reentrant. It is entered from DBRC in 31-bit addressing mode and must return to DBRC in 31-bit addressing mode. All parameters and data areas supplied to this exit routine by DBRC are located above the 16-megabyte line. In addition, load module DSPCINT0, in which the routine located, resides above the line. Note that due to the residency of DSPCINT0, unless you specify otherwise, GETMAIN will acquire storage above the 16-megabyte line when issued for the exit routine.

No further calls to the exit routine will occur if it terminates abnormally. DBRC will recover the termination and carry on normally thereafter.

Calling the Routine

Control is passed to the routine whenever there is a successful read, write, or modification made to **Copy 1** of the RECON data set. Header record extension control item changes, such as the RECON open flag, RESERVE count, or change copy field, also cause this routine to be called.

The IMS-supplied module is called only once. Thereafter, the module prevents further exit calls and their performance impacts from taking place. It is by replacing the IMS-supplied module with one that you write yourself that the exit routine is called continually.

Whenever a record of data is to be inserted, updated, deleted, or read, this routine is called and the journal updated **after** the call and/or change has been made to the RECON data set. For each insert, delete, and read call, the routine receives a copy of the inserted, deleted, or read record, respectively. For each update call, the routine receives a copy of the record as it appeared both before and after it was updated.

Any modifications to storage that this routine makes must be made to storage obtained by the routine, **not** to the data areas pointed to by DBRC/IMS or to those contained within the routine itself.

Each series of I/O accesses that DBRC makes to the RECON data set is indicated to the routine by a Begin Series call. When the series of I/O operations is complete, the routine receives a Terminate Series call.

Considering Performance

While this routine is running, the RECON data set is reserved so that no other jobs can access RECON records. To minimize the affect that this routine's execution has on your system's performance, you need to:

- Limit the I/O operations that the routine itself performs and simplify the routine's functions to make efficient use of processing time.
- Be sure that any resources needed solely by the routine (that is, those **not** needed by DBRC/IMS in general) are immediately available to MVS when DBRC is initialized and in control. You should therefore avoid operations that may put the routine, and therefore DBRC, in a prolonged wait state (for example, the ENQUEUE/DEQUEUE of resources that cannot be readily accessed by the routine or write to operator messages that require waiting for a reply).
- Make sure that the region size associated with DBRC is large enough to accommodate the routine and the functions it performs. If you need to enlarge the region size, increase it by:
  - The size of the assembled routine, plus
  - The maximum amount of storage that may be GETMAINed by the routine, plus
  - The total size of the buffers obtained by access methods for data sets used by the routine.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of a standard MVS parameter list. This list consists of a fullword with the high-order bit ON, indicating the last entry in the list. The remaining bits comprise the address of a data area containing information about the specific call.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to the calling RECON access routine.
15	Entry point of exit routine.

Description of Parameters

This routine receives the parameter list from the calling RECON access module at the first Begin Series call for a job. The parameter list points to the same data area for all subsequent calls for that job.

The data area pointed to by the parameter list is 16 words (64 bytes) long and starts on a fullword boundary. The last eight words of the list are free to be used by the exit routine and remain unchanged by DBRC/IMS after the first Begin Series call. They initially contain all zeros.

These are the contents of the data area at various exit points in the routine:

Begin Series exit call

Words	Meaning or Content
1	Eye-catcher “CEXT.”
2	A binary **1**, indicating a Begin Series call to this routine as a result of a RESERVE function having been performed on the RECON data set.
3-4	RESERVE sequence number from control record extension. This number is incremented by one in the control record extension each time DBRC completes a successful RESERVE of the RECON data set. This lets the routine know which series (within a sequence of series in a job) is about to be performed. When located within one IMS system in a shared RECON data set environment, this routine may not necessarily see an increment of one from one logical Begin Series call to the next.
5	Changed record count from the control record extension. The changed record count is a 32-bit logical value that can eventually wrap back to zero. This is the count as of the last DEQUEUE function performed by DBRC, or that value plus one if the last DBRC abends. A change to the RECON data set has occurred if an ENQUEUE sequence detects that the last DBRC abended. For more information about the changed record count, see Terminate Series exit call.
6-8	Not used.
9-16	Binary zeros on the first Begin Series call. Free to be used by the routine for this first call and all subsequent calls.

- **Insert exit call**

Words	Meaning or Content
1	Eye-catcher “CEXT.”

Words	Meaning or Content
2	A binary **3**, indicating an insert call to this routine as a result of a record having been inserted into the RECON data set.
3-4	Unchanged from the first Begin Series call.
5	Length of the record that's been inserted.
6	Address of the record that's been inserted.
7-8	Not used.
9-16	Free to be used by the routine as desired.

- **Update exit call**

Words	Meaning or Content
1	Eye-catcher "CEXT."
2	A binary **4**, indicating an update call to this routine as a result of a record having been updated on the RECON data set.
3-4	Unchanged from the first Begin Series call.
5	Length of the record pointed to by **Word 6**.
6	Address of a copy of the record as it appeared before the update.
7	Length of the replacement record.
8	Address of the replacement record.
9-16	Free to be used by the routine as desired.

- **Delete exit call**

Words	Meaning or Content
1	Eye-catcher ″CEXT″.
2	A binary **5**, indicating a delete call to this routine as a result of a record having been deleted from the RECON data set.
3-4	Unchanged form the first Begin Series call.
5	Length of the record that has been deleted.
6	Address of the record that has been deleted.
7-8	Not used.
9-16	Free to be used by the routine as desired.

- **Read exit call**

Words	Meaning or Content
1	Eye-catcher "CEXT."
2	A binary **6**, indicating a read call to this routine as a result of a record having been read from the RECON data set.
3-4	Unchanged from the first Begin Series call.
5	Length of the record that's been read.
6	Address of the record that's been read.
7-8	Not used.
9-16	Free to be used by the routine as desired.

- **Terminate Series exit call**

Words	Meaning or Content
1	Eye-catcher "CEXT."

Words	Meaning or Content
2	A binary **2**, indicating a Terminate Series call to this routine as a result of a DEQUEUE function having been performed on the RECON data set.
3-4	Unchanged from the Begin Series call.
5	Final changed record count as it now appears on the control record extension. The changed record count is a 32-bit logical value that can eventually wrap back to zero. The count is either the same as the Begin Series call value, or it is that value plus one if any change has been made, other than to the record extension itself, to the RECON data set since the Begin Series call. By monitoring the value of this counter between its value here and the next Begin Series exit call, you can detect changes made to the RECON data set by other occurrences of DBRC.
6-8	Not used.
9-16	Free to be used by the routine as desired.

Contents of Registers on Exit

Before returning to DBRC, the exit routine must restore all registers except register 15, which must contain one of the following return codes:

Return Code	Meaning
0	The exit routine is called.
non-0	No further calls to this exit routine are made.

Using the Sample RECON I/O Exit Routine

The sample RECON I/O exit routine records I/O activity that has taken place on the RECON data set. To minimize the performance impact that the routine's execution has upon DBRC, the routine spools its copy of RECON data records to a data set (specified by a DD statement with the name **DBRCDATA**) for later off-line processing outside the DBRC/IMS environment. Any data sets that your routine references need to be accessed by DD statements as well.

Chapter 25. Resource Access Security Exit Routine (DFSISIS0)

The Resource Access Security exit routine provides users without RACF a mechanism for checking authorization to IMS application group names (AGNs).

In this Chapter:

About This Routine

The Resource Access Security exit routine receives control when the dependent region goes to the control region for starting communication with the IMS control program or when a CCTL is connecting to IMS for DBCTL services. The exit routine is entered only once for each dependent region or CCTL. The routine can check the authorization of the dependent region or the CCTL to use the application group name. Each application group name represents a set of IMS resources defined through SMU. The resources are defined in the Security Maintenance Utility by the AGN control statement and data statements.

Related Reading: For details on specifying the AGN control and data statements and Resource Access security, see "Establishing IMS Security", in *IMS/ESA Administration Guide: System*.

For this exit to be executed, the AGNEXIT keyword can be specified in the SECURITY macro for system generation.

The sample Resource Access Security exit routine supplied with the system refuses authorization to all callers by returning a code of 8 in register 15 to ensure that the installation replaces the dummy exit routine supplied with the system.

Table 39 shows the attributes of the Resource Access Security exit routine.

Table 39. Resource Access Security Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL, and DBCTL
Naming convention	You must name this exit routine DFSISIS0.
Link editing	Follow the guidelines described in "Link-Editing the Routines" on page 5. You must write this exit routine as reentrant.
Including the routine	No special steps are needed to include this routine.
IMS callable services	This exit routine is not eligible to use IMS Callable Services.
Sample routine location	IMS.SVSOURCE (member name DFSISIS0)

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
0	A(SCD) Address of the SCD.
1	A(AGN) Address of the 8-byte Application Group Name (AGN).
3	A(A(AGN)) Address of the field that contains the address of the Application Group Name.
13	A(SAVE AREA) Address of the 18 fullword save area. This save area is pre-chained forward and backward. These pointers must not be modified.
14	Return address of IMS.
15	Entry point of exit routine.

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers except register 15, which must contain one of the following return codes:

Return Code	Meaning
0	AGN valid for this user.
8	AGN invalid for this user.

Chapter 26. System Definition Preprocessor Exit Routine (Input Phase) (DFSPRE60)

System Definition Preprocessor exit routine (Input Phase) can be used to alter, insert, or delete data from stage 1 input before the Preprocessor record scan occurs.

In this Chapter:

About This Routine

DFSPRE60 is a System Definition Preprocessor exit routine. This routine is entered following the reading each stage 1 input record.

Control passes to DFSPRE60 after each stage 1 input record is read from SYSIN, and also after each record (if any) is read from SYSLIB, but before any such records are scanned by the preprocessor. Stage 1 data is presented to DFSPRE60 exactly as read by the preprocessor. Data can be altered, inserted, or deleted during this exit routine phase. However, the alterations, insertions, or deletions are not passed to Stage 1.

Related Reading: For a description of the system definition preprocessor, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

You may also use this exit routine for construction of tables or for verification as required by installation practices. For example, if you want to check if transaction codes (which are also IMS command keywords) are accidentally defined, this routine can insert TRANSACT macros for each IMS command keyword. The results of changes appear in the listing produced by the preprocessor, unless the exit routine returns a return code of 4. However, no update of the original input is performed.

If this exit routine is used, you must specify Y as the first positional parameter in the parameter field of the EXEC card. The routine module must be named DFSPRE60, and it must reside on the library pointed to by the STEPLIB DD statement. If concatenation is used, the libraries are searched according to MVS rules.

Table 40 shows the attributes of the System Definition Preprocessor (Input Phase) exit routine.

Table 40. System Definition Preprocessor Exit Routine (Input Phase) Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL, and DBCTL (with modifications)
Naming convention	You must name this exit routine DFSPRE60.
Link editing	Follow the guidelines described in "Link-Editing the Routines" on page 5. This routine **must** be link-edited with RMODE=24; otherwise, an abend can occur.
Including the routine	No special steps are required to include this routine.
IMS callable services	IMS Callable Services are not applicable for use with this exit routine.

Table 40. System Definition Preprocessor Exit Routine (Input Phase) Attributes (continued)

Attribute	Description
Sample routine location	IMS.SVSOURCE (member name DFSPRE60)

Communicating with IMS

IMS communicates with the System Definition Preprocessor exit routine through the entry and exit registers.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of parameter list.
10	Address of vector table.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of exit routine.

Description of Parameters

The parameters are listed in parameter list format and vector table format.

Parameter List Format

```
A(INPUT) (or A(0) at end of file)
  INPUT: These stage 1 source statements reside on the data set
         pointed to by the DD statement SYSIN or the copy members
         from SYSLIB.
```

Vector Table Format

```
A(DFSPRE30)  -  Entry point to QUICKSORT routine
A(DFSPRE40)  -  Entry point to duplicate check routine
A(DFSPRE50)  -  Entry point to cross check routine
A(0)         -  Reserved for user
```

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers except for register 15, which must contain one of the following return codes:

Return Code	Meaning
00	Normal return; record is processed.
04	Do not process record. Record is not printed.
08	Return to exit routine. Pass control to this exit routine before reading another input record. The input buffer, as modified by the exit routine, is processed as an input record.
0C	Do not call this exit routine again. Record is processed.

Using the Sample System Definition Preprocessor Exit Routine

The input statements are scanned for a comment card indicating that TRANSACT macros are to be read from a user file, and passed to the preprocessor in the input buffer area whose address is passed on entry. While the user file is read, the exit routine returns a code of X'08', indicating that the preprocessor is to continue calling the exit routine, instead of reading input records from the SYSIN file.

When the end of file is reached on the user file, the exit routine returns a code of X'0C', indicating that the exit routine is not to be invoked again. The statements passed to the preprocessor are handled identically to the statements on the SYSIN file. However, these statements are not written out to the SYSIN file for later processing by Stage 1.

Chapter 27. System Definition Preprocessor Exit Routine (Name Check Complete) (DFSPRE70)

The System Definition Preprocessor exit routine (name check complete) DFSPRE70, can be used to build tables that contain resource names that have been cross-checked.

In this Chapter:

About This Routine

This exit routine is entered when all cross-checking of resource names is completed.

To track the changes between IMS system definitions, you may access all tables constructed by the preprocessor, and write them on files for input to a user program.

If this exit routine is used, you must specify Y as the second positional parameter in the parameter field on the EXEC card. The exit module, which must be named DFSPRE70, must reside on the library pointed to by the STEPLIB DD statement. If concatenated, the libraries are searched according to MVS rules. The processing in this exit routine does not affect the previous preprocessor results or error messages.

Table 41 shows the attributes of the System Definition Preprocessor exit routine (name check complete).

Table 41. System Definition Preprocessor Exit Routine (Name Check Complete) Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSPRE70.
Link editing	Follow the guidelines described in “Link-Editing the Routines” on page 5.
Including the routine	No special steps are required to include this routine.
IMS callable services	IMS Callable Services are not applicable for use with this exit routine.
Sample routine location	IMS.SVSOURCE (member name DFSPRE70) In the sample routine, the source name tables are written out for later processing by user programs. The end of each exit routine is indicated by the insertion of high values (X'FF').

Communicating with IMS

IMS communicates with the System Definition Preprocessor exit routine through the entry and exit registers.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of parameter list.
10	Address of vector table.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of exit routine.

Description of Parameters

These are the addresses and sizes of tables of resource names. The contents of the count field (fullword) determine the usefulness of the table field. If the count field is zero, the contents of the table field are not guaranteed and should be ignored. If the count field is nonzero, the table field contains the address of the table of resource names.

Parameter List:

```
A(Table1)  -  DBD names
A(Count1)  -  Number of entries in Table1
A(Table2)  -  PSB names
A(Count2)  -  Number of entries in Table2
A(Table3)  -  Transaction codes
A(Count3)  -  Number of entries in Table3
A(Table4)  -  MSNAME linknames
A(Count4)  -  Number of entries in Table4
A(Table5)  -  VTAM node names
A(Count5)  -  Number of entries in Table5
A(Table6)  -  LTERM names
A(Count6)  -  Number of entries in Table6
A(Table7)  -  Subpool names
A(Count7)  -  Number of entries in Table7
A(Table8)  -  Routing codes
A(Count8)  -  Number of entries in Table8
A(Table9)  -  Physical link names
A(Count9)  -  Number of entries in Table9
A(Table10) -  Remote system VTAM node names
A(Count10) -  Number of entries in Table10
A(Table11) -  MSLINK partner IDs
A(Count11) -  Number of entries in Table11
```

Vector Table Format:

```
A(DFSPRE30)  -  Entry point to QUICKSORT routine
A(DFSPRE40)  -  Entry point to duplicate check routine
A(DFSPRE50)  -  Entry point to cross check routine
A(0)         -  Reserved for user
```

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers. Register 15 can contain a return code, but the preprocessor ignores it.

Chapter 28. User Message Table (DFSCMTU0)

Although there are IMS system message tables containing messages that IMS returns to edit and exit routines, these messages may not be appropriate for your installation's needs. If this is the case, you can create your own messages and list them in your own message table.

In this Chapter:

About This Table

IMS assigns the prefix of the user text from the message table with DFSUxxx, where *xxx* is the message number. You can then use this message table with the following user edit and exit routines:

- Command Authorization exit routine (DFSCCMD0)
- Front-End Switch exit routine (DFSFEBJ0)
- Global Physical Terminal edit routine (DFSGPIX0)
- Logoff exit routine (DFSLGFX0)
- Message Switching Input edit routine (DFSCNTE0)
- Input Message Segment edit routine (DFSME127)
- Physical Terminal Input edit routine (DFSPIXT0)
- Queue Space Notification exit routine (DFSQSPC0)
- Sign On/Off Security exit routine (DFSCSGN0)
- Sign-On exit routine (DFSSGNX0)
- Sign-Off exit routine (DFSSGFX0)
- Transaction Code Input edit routine (DFSCSMB0)

Restriction: This table cannot be used in a DBCTL environment.

Writing a Routine to Use This Table

In order for a routine to use the messages you've placed in the message table, you'll need to choose a key that represents a message number in the table. In the case of the Queue Space Notification exit routine (DFSQSPC0) and the Sign-On exit routine (DFSSGNX0), the **negative** value of the message key needs to be placed in register 15 on return from the routine. For the other exit routines listed in the previous section, the **positive** value of the message key needs to be placed in register 1 on return from the routine along with a specific return code in register 15.

Exception: The Front-End Switch exit routine (DFSFEBJ0) is an exception to this. Refer to the descriptions of each exit routine for more information.

Including the Table During System Definition

Before Stage 1 of IMS system definition, you need to specify OPTIONS=(...USERMSGS,...) in the COMM macro.

Related Reading: For more information, see the COMM macro statement description in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Naming the Table

You must name the message table module **DFSCMTU0**, assemble it, and place it in the operating system partitioned data set defined by the USERLIB= operand of the IMSGEN macro.

Related Reading: For details about this, see the IMSGEN macro statement description in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Formatting the Table

The format of your message table (DFSCMTU0) must be as follows:

- The table must start with the instruction BALR 15,14.
- Message numbers range from 1 to (and including) 999, in ascending sequence.
- The maximum size for the text of each message is 100 characters, If the message text exceeds 78 characters, it will be truncated if it is sent to a 3270 terminal that is formatted using IMS-supplied default formats.
- The message length must be an even value (otherwise, an erroneous character may appear following the last character of the text).
- Exclude device control characters from the text of your messages. IMS always adds NEW LINE (NL) control characters to the beginning and end of each message. For messages processed by the Message Format Service (MFS), the device control characters will be changed to X'00' for a 3270 display and to X'40' for other devices.
- Each message entry must start on a half-word boundary. The entry format is:

```
label   DC  H'message number'
        DC  AL2 (entry length including number
            and length fields)
        DC  C'message text of even length'
```

- An entry with message number X'7FFF' signals the end of the message table.

Using IMS Callable Services

IMS Callable Services are not applicable for use with this table.

Sample User Message Table and Routine

This example shows how you would use user message tables to change the text of the messages issued by a revised version of the Queue Space Notification exit routine.

Sample Table

The following User Message Table contains the messages specified by an IMS user and has been included in the IMS system:

```
DFSCMTU0 CSECT
*                                USER MESSAGE TABLE FOR USER QUEUE
*                                SPACE NOTIFICATION EXIT EXAMPLE.

         BALR  15,14
M013     DC    H'513'            QMGR0
         DC    AL2(M014-M013)
         DC    C'RECORDS IN QBLKS DATASET EXCEED UPPER THRESHOLD '
M014     DC    H'514'            QMGR0
         DC    AL2(M015-M014)
         DC    C'RECORDS IN SMSGQ DATASET EXCEED UPPER THRESHOLD '
M015     DC    H'515'            QMGR0
         DC    AL2(M016-M015)
         DC    C'RECORDS IN LMSGQ DATASET EXCEED UPPER THRESHOLD '
M016     DC    H'516'             QMGR0
         DC    AL2(M017-M016)
         DC    C'RECORDS IN QBLKS DATASET BELOW LOWER THRESHOLD'
M017     DC    H'517'             QMGR0
         DC    AL2(M018-M017)
         DC    C'RECORDS IN SMSGQ DATASET BELOW LOWER THRESHOLD'
M018     DC    H'518'             QMGR0
         DC    AL2(M999-M018)
         DC    C'RECORDS IN LMSGQ DATASET BELOW LOWER THRESHOLD'
M999     DC    X'7FFF'
         END   ,
```

Sample Routine

In this sample routine, the IMS-supplied exit routine (DFSQSPC0) has been replaced by a modified version of the routine that a user has written. The user-modified DFSQSPC0 has the following characteristics:

1. Existing IMS message equates have been replaced by user equates.
2. The list of messages used by the routine code has been changed to refer to the user messages.
3. Load Negative Register (LNR) instructions have been added to store the negative of the user message key in register 15 before returning to the caller of DFSQSPC0. This causes IMS to look in the User Message Table (DFSCMTU0) rather than the system tables for the text of the message.

The following is the modified Queue Space Notification exit routine, DFSQSPC0:

```
***********************************************************************
*                                                                     *
*              M O D U L E       P R O L O G                          *
*                                                                     *
***********************************************************************
*                                                                     *
* MODULE NAME: DFSQSPC0                                               *
*                                                                     *
* DESCRIPTIVE NAME: SAMPLE USER QUEUE SPACE NOTIFICATION EXIT          *
*                                                                     *
* FUNCTION:                                                           *
*                                                                     *
*    INTERROGATES NUMBER OF RECORDS CURRENTLY IN USE FOR A            *
*    DATASET AND DETERMINES WHETHER OR NOT TO DISPLAY                 *
*    THRESHOLD MESSAGES (SEE OUTPUT)                                  *
*                                                                     *
* NOTES:                                                              *
*                                                                     *
*    RESTRICTIONS:                                                    *
*                                                                     *
*    DFSQSPC0 MUST NOT IWAIT. THERE IS ONLY ONE PARAMETER AREA        *
*    (IN QPOOL), HENCE THE QUEUE MANAGER MUST NOT IWAIT BETWEEN       *
*    THE TIME IT SETS UP THE PARAMETER LIST AND THE TIME IT NO        *
*    LONGER NEEDS IT, FOLLOWING INVOCATION OF THE EXIT (DFSQSPC0)     *
*                                                                     *
*    IN ORDER TO UPDATE THE "IN USE" COUNT WITHOUT FIRST ZEROING      *
*    THE HIGH ORDER BYTE THE HIGH ORDER BIT OF THE FLAG BYTE MUST     *
*    ALWAYS BE 0.                                                     *
*                                                                     *
*    DEPENDENCIES: NONE                                               *
*                                                                     *
*    REGISTER CONVENTIONS: STANDARD IMS                               *
*                                                                     *
*    MODULE TYPE:                                                     *
*                                                                     *
*    IMS DC - QUEUE MANAGER EXIT (MAY BE REPLACED BY USER EXIT)       *
*                                                                      *
*    ATTRIBUTES: REENTRANT                                             *
*                                                                      *
* ENTRY POINT: DFSQSPC0                                                *
*                                                                      *
*    PURPOSE: SEE FUNCTION                                             *
*                                                                      *
*    LINKAGE: BALR R14,R15 FROM DFSQMGR0 WHENEVER AN LRECL (DRRN)      *
*    IS ASSIGNED OR FREED                                              *
*    NOTE: IN ORDER TO REDUCE THE NUMBER OF INSTRUCTIONS IN THE        *
*    EXIT ONLY THE WORK REGISTERS (4 AND 5) ARE SAVED AND RESTORED     *
*                                                                      *
* INPUT:                                                               *
*                                                                      *
*    R0  =  DATASET INDICATOR                                          *
*           00 IF QBLKS                                                *
*           04 IF SMSGQ                                                *
*           08 IF LMSGQ                                                *
*                                                                      *
*    R2  =  POINTER TO PARAMETER LIST...                               *
*           1ST WORD, 1ST BYTE                                         *
*                    = FLAG CONTAINING BIT (X'40') THAT INDICATES      *
*                      WHETHER (ON) OR NOT (OFF) # OF RECORDS IN       *
*                      USE EXCEEDED THE UPPER THRESHOLD BUT HAS        *
*                      NOT YET DROPPED BELOW THE LOWER THRESHOLD       *
*           1ST WORD, 2ND-4TH BYTE                                     *
*                    = # OF RECORDS CURRENTLY IN USE                   *
*           2ND WORD = MAX # OF RECS ASSIGNABLE BEFORE SHUTDOWN        *
*                                                                      *
*    R10 = SCD ADDRESS                                                 *
*                                                                      *
*    R14 = RETURN ADDRESS                                              *
*                                                                      *
*    THE UPPER AND LOWER THRESHOLD VALUES ARE OBTAINED FROM SCD        *
*    FIELDS SCDQTU AND SCDQTL. THE QUEUE MANAGER INITIALIZATION        *
*    MTHE JCL USED TO BRING UP THE CONTROL REGION.                     *
*    CORRESPONDING FIELDS IN RGPARMS. RGQTU AND RGQTL ARE              *
*    1) IMS DEFAULTS (75% AND 60%), 2) USER DEFAULTS ESTABLISHED       *
*    BY SPECIFYING VALUES FOR QTU AND QTL IN THE DFSPBxxx MEMBER       *
*    OR 3) EXEC PARAMETER VALUES FOR QTU AND QTL ON                    *
*    THE JCL USED TO BRING UP THE CONTROL REGION.                      *
*                                                                      *
* OUTPUT:                                                              *
*                                                                      *
*    R0, R2, AND R10 ARE UNCHANGED                                     *
*    R15 = RETURN CODE                                                 *
*        = 0 IF NO THRESHOLDS PASSED                                   *
*        = ONE OF THE FOLLOWING MESSAGE KEYS IF THRESHOLD PASSED:      *
*                                                                      *
DFS513   EQU  513   RECS IN QBLKS EXCEED UPPER THRESHOLD               *
*          IF QBLKS UPPER THRESHOLD EXCEEDED (BIT X'40' IN             *
*          PARAMETER LIST FLAG BYTE WILL BE TURNED ON)                 *
*                                                                      *
DFS514   EQU  514   RECS IN SMSGQ EXCEED UPPER THRESHOLD               *
*          IF SMSGQ UPPER THRESHOLD EXCEEDED (BIT X'40' IN             *
*          PARAMETER LIST FLAG BYTE WILL BE TURNED ON)                 *
*                                                                      *
DFS515   EQU  515   RECS IN LMSGQ EXCEED UPPER THRESHOLD               *
*          IF LMSGQ UPPER THRESHOLD EXCEEDED (BIT X'40' IN             *
*          PARAMETER LIST FLAG BYTE WILL BE TURNED ON)                 *
*                                                                      *
DFS516   EQU  516   RECS IN QBLKS DATASET BELOW LWR THRESHOLD          *
*          IF QBLKS LOWER THRESHOLD PASSED (BIT X'40' IN               *
*          PARAMETER LIST FLAG BYTE WILL BE TURNED OFF)                *
*                                                                      *
DFS517   EQU  517   RECS IN SMSGQ DATASET BELOW LWR THRESHOLD          *
*          IF SMSGQ LOWER THRESHOLD PASSED (BIT X'40' IN FLAG          *
*          PARAMETER LIST FLAG BYTE WILL BE TURNED OFF)               *
*                                                                     *
DFS518   EQU  518   RECS IN LMSGQ DATASET BELOW LWR THRESHOLD         *
*          IF LMSGQ LOWER THRESHOLD PASSED (BIT X'40' IN              *
*          PARAMETER LIST FLAG BYTE WILL BE TURNED OFF)               *
*                                                                     *
*    NOTE: IF DFSQSPC0 IS REPLACED BY A USER EXIT THE USER            *
*          MESSAGE NUMBERS MUST BE RETURNED IN R15 AS THE             *
*          NEGATIVE OF THE POSITIVE MESSAGE NUMBER (LNR).             *
* NORMAL EXIT: SEE OUTPUT                                             *
*                                                                     *
* ERROR EXIT: NONE                                                    *
*                                                                     *
* EXTERNAL REFERENCES: NONE                                           *
*                                                                     *
* CHANGE ACTIVITY: SEE CHANGEID                                       *
*                                                                     *
***********************************************************************
         EJECT
***********************************************************************
*
*    PSEUDO CODE
*
***********************************************************************
*
*
*    IF THE BIT IN THE PARAMETER FLAG INDICATING # OF RECORDS
*         EXCEEDED UPPER THRESHOLD IS ON
*
*         THEN
*         IF # OF RECORDS CURRENTLY IN USE HAS DROPPED BELOW LOWER
*         THRESHOLD (60% OF MAXIMUM # OF ASSIGNABLE RECORDS BEFORE
*         SHUTDOWN)
*
*              THEN
*              TURN OFF BIT IN PARAMETER FLAG INDICATING # OF RECORDS
*              EXCEEDED UPPER THRESHOLD.
*              SET R15 = KEY FOR MESSAGE INDICATING DATASET OK AGAIN.
*
*              ELSE
*              NULL
*
*         ELSE
*         IF # OF RECORDS CURRENTLY IN USE > UPPER THRESHOLD (75% OF
*              MAXIMUM # OF ASSIGNABLE BYTES BEFORE SHUTDOWN)
*
*              THEN
*              SET BIT IN PARAMETER FLAG INDICATING # OF RECORDS
*              EXCEEDED UPPER THRESHOLD.
*              SET R15 = KEY FOR MESSAGE INDICATING UPPER THRESHOLD
*              EXCEEDED.
*
*              ELSE
*              NULL
*
*    RETURN TO CALLER.
*
***********************************************************************
         EJECT
DFSQSPC0 CSECT
         CHANGEID NAME=DFSQSPC0&SYSDATE,BASE=R12,LINKAGE=IMS,          X
               SAVE=(4,,5,,12,)
         CHANGEID IDEND=YES
*
         USING PARM,R2       ADDRESSABILITY TO PARM AREA.
         USING SCD,R10       ADDRESSABILITY TO SCD.
*
```

```
         TM    PFLAG,PFEXCD  EXCEEDED UPPER THRESHOLD?
         BZ    QSPC100       NO... CONTINUE.
*
         L     R5,PMAX       CALCULATE LOWER THRESHOLD
         M     R4,SCDQTL     (60% OF MAXIMUM NUMBER
         D     R4,=F'100'    OF RECORDS ASSIGNABLE).
*
         L     R4,PINUSE     GET FLAG + IN USE COUNT.
         LA    R4,0(,R4)     GET RID OF FLAG.
         CR    R5,R4         LOWER THRESHOLD : # CRNTLY IN USE
         BNH   RETURN        BR IF LOWER THRESHLD <= CUR IN USE.
*
         NI    PFLAG,X'FF'-PFEXCD TURN OFF EXCEEDED FLAG.
         LR    R15,R0        SET UP
         SRL   R15,1         INDEX.
         LH    R15,MSGTBL1(R15)   GET APPROPRIATE MESSAGE KEY.
         LNR   R15,R15       INDICATE USER MESSAGE KEY.
         B     RETURN1
*
         L     R5,PMAX       CALCULATE UPPER THRESHOLD
         M     R4,SCDQTU     (75% OF MAXIMUM NUMBER
         D     R4,=F'100'    OF RECORDS ASSIGNABLE).
*
         L     R4,PINUSE     GET FLAG + IN USE COUNT.
         LA    R4,0(,R4)     GET RID OF FLAG.
         CR    R5,R4         UPPER THRESHOLD : # CURNTLY IN USE
         BNL   RETURN        BR IF UPPER THRESHLD >= CUR IN USE.
*
         OI    PFLAG,PFEXCD  SHOW CURRENT IN USE EXCEEDED MAX.
         LR    R15,R0        SET UP
         SRL   R15,1         INDEX.
         LH    R15,MSGTBL2(R15)   GET APPROPRIATE MESSAGE KEY.
         LNR   R15,R15       INDICATE USER MESSAGE KEY.
         B     RETURN1
*
RETURN   EQU   *
         SR    R15,R15       RC FOR NO THRESHOLDS PASSED.
*
RETURN1  EQU   *
         LEAVE RC=(15),RESTORE=(4,,5,,12,)
         EJECT
***********************************************************************
*
*        MESSAGES RETURNED WHEN THRESHOLDS PASSED
*
***********************************************************************
*
MSGTBL1  DS    0H
         DC    AL2(DFS516)   RECS IN QBLKS BELOW LOWER THRESH
         DC    AL2(DFS517)   RECS IN SMSGQ BELOW LOWER THRESH
         DC    AL2(DFS518)   RECS IN LMSGQ BELOW LOWER THRESH
MSGTBL2  DS    0H
         DC    AL2(DFS513)   RECS IN QBLKS EXCEED UPPER THRESH
         DC    AL2(DFS514)   RECS IN SMSGQ EXCEED UPPER THRESH
         DC    AL2(DFS515)   RECS IN LMSGQ EXCEED UPPER THRESH
         EJECT
         LTORG
         EJECT
         REQUATE
         PRINT NOGEN
         ISCD
         PRINT GEN
         EJECT
         DFSPARM
         END
```

Part 4. Transaction Manager Exit Routines

Chapter 29. Type 1 Automated Operator Exit Routine (DFSAOUE0)

You can write two types of automated operator (AO) exit routines. The AO exit routine described in this chapter is called a type 1 AO exit routine. It can be used in the DB/DC and DCCTL environments.

The other AO exit routine (DFSAOE00) is called a type 2 and can be used in the DB/DC, DCCTL, and DBCTL environments.

If both DFSAOUE0 and DFSAOE00 are provided in a DB/DC or DCCTL environment, DFSAOE00 is called first. DFSAOE00 determines which exit routine will handle the message, command, or command response.

Related Reading:

- DFSAOE00 is described in “Chapter 11. Type 2 Automated Operator Exit Routine (DFSAOE00)” on page 119.
- For an overview and comparison of the two AO user exit routines, see *IMS/ESA Operations Guide*.

In this Chapter:

About This Routine

This AO exit routine is called continuously for system messages destined for the master terminal, operator-entered commands, and command responses. The AO exit routine intercepts these messages before IMS sends the system message, executes the command, or sends the command response. System messages destined for terminals other than the master terminal operator (MTO) and certain commands and command responses do not cause IMS to call this exit routine. (For more information, see “Types of Messages Not Passed to This Routine” on page 232.)

You can write the exit routine to handle both single and multisegment messages, and to perform the following functions:

- Ignore selected segments or an entire message.
- Send a copy of a system message, command, or command response to an alternate destination.
- Send a new message to an alternate destination for a system message, command, or command response.
- Change a system message.
- Change a system message and send a copy to an alternate destination.
- Change a copy of a command or command response and send the copy to an alternate destination.
- Delete a system message.
- Delete a system message and send a copy to an alternate destination.
- Request the edited command buffer (when the input is a command).

DFSAOUE0

Table 42 shows the attributes for DFSAOUE0.

Table 42. Automated Operator Exit Routine Attributes (DFSAOUE0)

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSAOUE0.
Link editing	DFSAOUE0 is a stand-alone, 31-bit module that you must provide. **Recommendation:** It is recommended, but not required, that the module be reentrant. You must manually link edit the routine with DFSCSI00 to include the routine. See "Link-Editing the Routines" on page 5 for general guidelines on link editing exit routines.
Including the routine	You specify DFSAOUE0 by linking it in the IMS.RESLIB concatenation as a stand-alone module. DFSAOUE0 is then loaded and called. If you specify both DFSAOUE0 and DFSAOE00 (the other AO exit routine), both are loaded. DFSAOE00 is called first and can either process the message, command, or command response, or it can return a code indicating DFSAOUE0 should be called to do the processing instead.
IMS callable services	DFSAOUE0 can use callable services for storage and control block functions. To use callable services, issue an initialization call (DFSCSII0) to get the callable services token and a parameter list in which to build the function-specific parameter list for the callable service you want to use. Use the ECB found in register 9 for the DFSCSII0 call. For guidelines on using callable services, see "Using IMS Callable Services" on page 8.
Sample routine location	IMS.TMSOURCE The AO exit routine can work with an AO application. The exit routine can insert a message to an alternate destination that is an AOI transaction without using an AO application. (For more information, see "Using the Sample AO Application (UETRANS)" on page 443.) The AO exit routine and the AO application used together serve as an example of how to use AOI.

Restrictions

The exit routine can change or delete system messages only. It can modify the copy of the system message that the original destination (the master terminal) receives. It can also modify the copy that an alternate destination receives. The exit routine **cannot** change or delete the original command or command response. It can modify the copy of a command or command response that is destined for an alternate destination, but it **cannot** change the copy that the primary destination receives.

Some transactions must reside on the same IMS subsystem as DFSAOUE0 to process correctly. If your installation uses shared queues, define these local transactions as SERIAL to guarantee that they are processed on the local IMS subsystem. A transaction that is not defined as SERIAL can be processed on any IMS subsystem that has that transaction defined.

How This Routine Processes Messages

Figure 13 shows an AO exit routine intercepting a command entered from a master terminal.

1. The command is entered.

2. The command controller passes a copy of the command to the exit routine before executing the command. The exit routine can send a copy of the command to any destination (LTERM or transaction).
3. The exit routine returns to the command controller, where the command is executed.
4. When the response to the command is returned to the command controller, a message is generated for the master terminal.
5. Before the message is sent, the exit routine receives control and can route a copy of the message to any destination (LTERM or transaction).
6. The message is then sent to the master terminal.

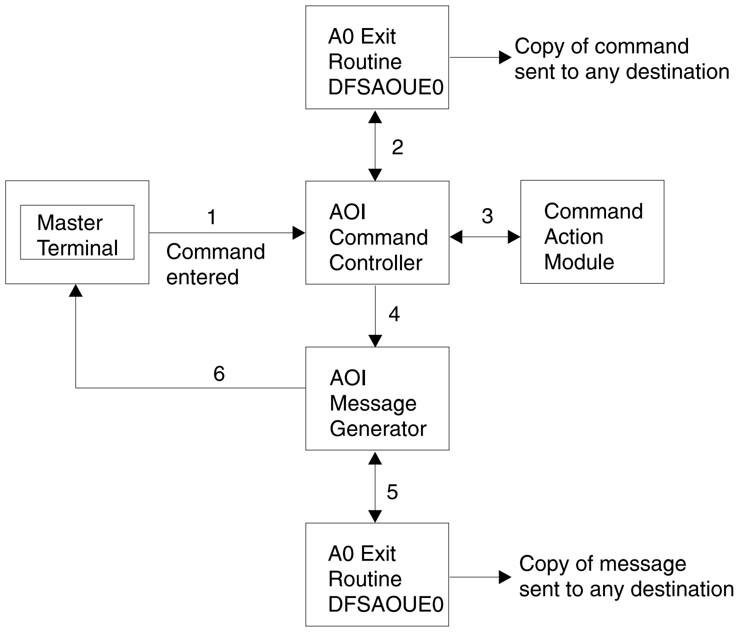


Figure 13. Processing When a Command is Entered at the Terminal

Figure 14 shows an exit routine processing a system message destined for the master terminal. When a system message is generated, the exit routine receives a copy of the message before the message is sent to the master terminal. The exit routine can route a copy of the message to any destination. It can alter or delete any segment of the message.

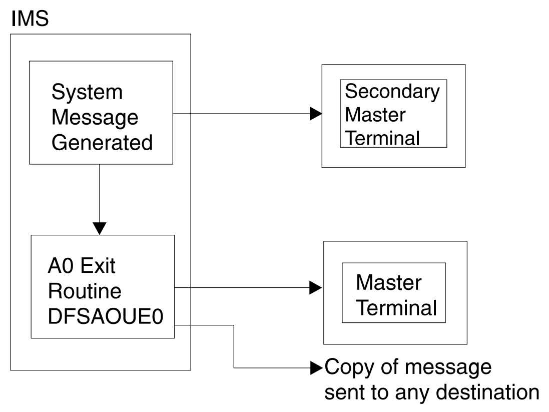


Figure 14. Processing When a System Message Is Generated

A sample exit routine is described in Table 42 on page 228. The contents of the User Exit Header Block (UEHB) are described in “User Exit Header Block (UEHB)” on page 247.

Related Reading: If you use the exit routine with AOI transactions, refer to “Part 5. Automated Operator Program Interface” on page 433 for more information on AOI.

Types of Messages Passed to This Routine

The following sections contain information about which messages are passed to the exit routine. IMS passes a copy of commands and command responses, and system messages destined for the master terminal. (For the format of a copy, see “Format of Message Segment Copies” on page 233.)

A message that is passed to the exit routine can contain multiple segments. For more information on how to write the exit routine to handle multisegment messages, see “Supporting Multisegment Messages” on page 233.

System Messages

A system message is a DFS message that is not a direct (synchronous) response to a command. IMS passes to the exit routine system messages that are destined for the master terminal. (If the system message is destined for the secondary master terminal or MVS system console, IMS does not pass the exit routine a copy of the message.) Bit UEH1CPYP in the UEHFLG1 flag field of the UEHB indicates that the input to the exit routine is a system message. In most cases, the first system message that causes IMS to call the exit routine is the DFS994 checkpoint message.

Related Reading: For more information, see *IMS/ESA Messages and Codes*.

While IMS passes a system message destined for the master terminal, it can also send a copy of this message to the secondary master terminal or the MVS system console. IMS sends this copy before passing the original message to the master terminal. The exit routine can only change the original message destined for the master terminal. The copy that the secondary master terminal or MVS system console receives does not reflect any changes the exit routine makes to the original message.

Most system messages are single-segment messages. Some system messages are multisegment messages (such as DFS802, DFS970, DFS2503, and DFS3222).

Commands

IMS passes the exit routine a copy of each IMS command entered except:

- Internally generated commands.
- Commands issued by a CMD or ICMD call from an AO application.
- `/FORMAT`
- `/LOOPTEST`
- `/MSVERIFY`
- `/RELEASE`
- `/NRESTART`
- `/ERESTART`

IMS passes the command after basic edit and optional editing routines have had a chance to modify it. This modified input can contain carriage control characters.

Commands with Network-Qualified LU Names: If you use network-qualified LU names at your installation, the LU name can be 17 bytes long. For IMS commands, the network-qualified LU names must be enclosed in single quotes (for example, 'NETID.LUNAME').

If an IMS command with the network-qualified LU name is passed to the AO exit routine, IMS modifies the network-qualified LU name in the input command before the command is passed to the AO exit routine. The single quotes around the network-qualified LU name are replaced with blanks, and the period separating the network-identifier and the LU name is replaced with a colon.

Example:A /DISPLAY command with a network-qualified LU name entered at the terminal as:

```
/DISPLAY LUNAME 'NETWORK1.LUNAME1' LUNAME2 INPUT.
```

is passed to the AO exit routine or logged to the secondary master as:

```
/DISPLAY LUNAME  NETWORK1:LUNAME1  LUNAME2 INPUT.
```

Command Responses

IMS passes a copy of command responses to the exit routine. A command response is a copy of the original response that IMS sent to the terminal that entered the command. Any asynchronous system message that IMS produces as a result of a command is **not** considered a command response, and is passed to the exit routine only if its destination is the master terminal (as is the case with all system messages that IMS passes to the exit routine). The exit routine can request that the edited command buffer be made available on the last entry by setting a flag in the UEHB. (For information on the use and format of this buffer, see “Viewing the Edited Command Buffer” on page 234.)

To receive a command response, the exit routine must handle multisegment messages and check for subsequent segments; this is because the first segment of a command response is considered the second segment of a command, even if the response has only one segment. Responses to the `/DISPLAY`, `/RDISPLAY` and `/RMxxxx` commands are multiple segment responses.

Changes the Command Editor Makes

The command editor translates certain control characters in any commands you enter from a terminal. You need to accommodate this translation when writing your exit routine.

The translation is as follows:

Table 43. Translation of Control Characters in Commands

From	To
X'14' Restore	X'5D' Right Parenthesis
X'15' New Line	X'40' Blank
X'24' Bypass	X'4D' Left Parenthesis
X'40' Blank	X'40' Blank
X'4B' Period	X'4B' Period
X'4D' Left Parenthesis	X'4D' Left Parenthesis
X'5D' Right Parenthesis	X'5D' Right Parenthesis
X'60' Dash	X'60' Dash
X'6B' Comma	X'6B' Comma
X'6D' Dash	X'40' Blank
X'7E' Equal	X'40' Blank

Types of Messages Not Passed to This Routine

IMS does not pass all system messages, operator-entered commands, and command responses to this exit routine. The following are messages that IMS does **not** pass to the exit routine:

- Messages resulting from a /BROADCAST command, other than the command and the initial response
- Messages associated with the /FORMAT, /LOOPTEST, /MSVERIFY, and /RELEASE commands and their responses
- Copies of system messages destined for the secondary master terminal or the MVS system console
- Copies of message switches, messages inserted by application programs, or messages resulting from the /BROADCAST command
- All system messages, commands, and command responses **if** message queues are unavailable (which is possible when initializing, restarting, or shutting down IMS)

Single- and Multisegment Messages

The exit routine cannot determine from the first segment of a message whether it is a multisegment message or not. You can determine which messages are single-segment messages, and write the exit routine so that IMS only calls it once. This will enable your installation to avoid any overhead incurred when the exit routine is written to always test for subsequent segments. To handle those messages that are multisegment messages or for which you can't predetermine the number of segments, you can write the exit routine to request all remaining segments.

If you write the exit routine so that it doesn't differentiate between single and multisegment messages and always checks for remaining segments, IMS always calls it at least twice. A message segment can accompany the last entry to the exit routine when the message is a multisegment message. No segment is presented to the exit routine when the message is a single-segment message.

Supporting Multisegment Messages

Although most messages contain only one segment, some messages are multisegment messages. System messages DFS970 and the response to a `/DISPLAY` command are examples of multisegment messages. Even if a command response is a single-segment response, the exit routine must be written to handle multisegment messages; this is because a command always precedes a command response. If the exit routine does not check for subsequent segments, IMS does not pass the segments containing the command response.

You can write the exit routine so that IMS calls it for each segment of a message. If you write the exit routine to request the remaining segments, the exit routine is called at least twice for each message, even if the message has only one segment. In this case, the last entry to the exit routine is not accompanied by a segment; this is because the message is a single segment. If the message is a multisegment message, the exit routine is called for the subsequent segments. The exit routine must test for a segment the last time it is entered.

For subsequent entries to the exit routine for a multisegment message, bit UEH1SEG is set in the UEHBFLG1 field of the UEHB when another segment is being presented. UEHCPYBF points to the next segment of the message.

The exit routine cannot necessarily tell which segment belongs to which message; this is because the presentation of segments associated with any one message can be interspersed with segments associated with a different message. The UEHB is unique for each message, and you can use the UEHURSVD field to keep track of which message IMS presents to the exit routine. (For more information, see “User Exit Header Block (UEHB)” on page 247.)

Format of Message Segment Copies

IMS uses the UEHB to pass a copy of the message segment to the exit routine and places the address of that message segment in the UEHCPYBF field of the UEHB. The format of the copy of a system message, operator-entered command, or command response is:

LL	ZZ	message text	CR	work area

The message segment copy contains the following fields:

LL	2-byte field containing the length of the message on first entry to the exit routine, not including the length of the 20-byte work area. (If your exit routine deletes or changes a message, or uses the work area, it must update this length field.)
ZZ	2-byte field reserved for IMS.
message text	System message: The first segment of the message text begins with the DFS*xxxx* number, indicating which message caused IMS to call the exit routine. The message number is followed by the text of the system message. If it is a multisegment message, the remaining segments contain additional text, but do not contain the DFS*xxxx* message number. Command: The message text is one segment long and begins with the delimiter '/', followed by the command. Command response: The command response is usually a DFS*xxxx* message or one segment of a multisegment command.

CR	Optional 1-byte field containing carriage control characters (for example, X'15'). Input commands do not include a carriage control character. If the CR field is included, one byte is included in LL.
work area	A 20-byte work area added to the end of the system message, command, or command response; the exit routine can use this work area to communicate with the alternate destination for that segment.

Viewing the Edited Command Buffer

IMS expands certain commands and places this expanded view into the edited command buffer. You can examine this buffer by setting the appropriate exit registers.

One occasion for examining the buffer is when command processing exceptions occur, indicated by the DFS058 XXX COMMAND COMPLETED EXCEPT message. When a LINE, LINK, NODE, or PTERM keyword is used with inclusive or range parameters, or when a LINE, LINK, PTERM, or SUBSYS keyword is used with the ALL parameter, IMS expands the command in the edited command buffer to include the actual resource names or numbers, except for the `/BROADCAST` command. The PTERM ALL keywords are only expanded for the `/PSTOP`, `/PURGE`, `/RSTART`, `/START`, `/STOP`, and `/MONITOR` commands. When a NODE, LTERM, or USER keyword is used with a generic parameter and exceptions occur, IMS expands the edited command buffer with up to ten of the specific resource names that are invalid and that match the generic parameter.

Only parameter passwords (as in the `/IAM` command) are shown in the edited command buffer; command passwords are not shown.

Figure 15 shows the format of the edited command buffer.

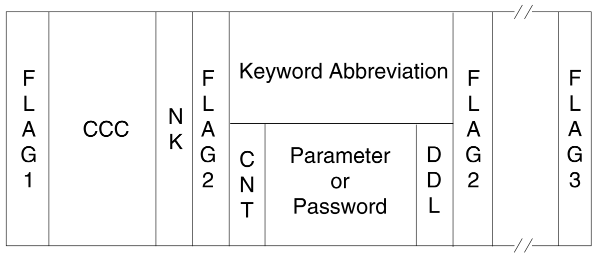


Figure 15. Edited Command Buffer

The fields contain the following:

FLAG1: Field containing one of the following flags:
- X'FE'—Beginning of the edited command.
- X'FC'—An error was found in a parameter and this flag was set by the command action modules (except for `/IAM` command processing).

CCC: First 3 bytes of command.

NK: Hexadecimal value of the number of keywords in the edited command buffer.

FLAG2: Field containing one of the following flags:
- X'FC'—Parameter that follows is in error.

- X'FF'—3-byte keyword abbreviation follows.
- X'FE'—Count (CNT) field and parameter follow.
- X'('—Count (CNT) field and password follow.

Keyword Abbreviation: Refer to DFSCKWD0 to obtain the abbreviation. In some cases, the abbreviation is the first three characters of the keyword.

CNT: Number of characters in the parameter or password that immediately follow the CNT. (This CNT field is a 1-byte binary field.)

Parameter or Password: Parameter exactly as entered from the terminal.

DDL: Delimiter entered after the parameter or password. If the ALL parameter is expanded to individual parameters, the delimiter is X'80'. If the parameter is generic, the delimiter is X'10'.

FLAG3: Period indicating the end of the command.

Activating This Routine

DFSAOUE0 is activated after IMS restart is complete. DFSAOUE0 is activated for each system message, command, and command response, as explained in previous sections.

When IMS shutdown processing has begun, DFSAOUE0 is disabled and no longer receives control.

Communicating with IMS

IMS communicates with this exit routine through the entry and exit registers, and the user exit header block (UEHB). IMS creates a UEHB for **each** message and passes it to the exit routine every time the exit routine is called for that message. Your exit routine can use the UEHURSVD field in the UEHB to store information about the message between each call to the exit routine for that message. The UEHB is freed and any values that were previously saved are lost when processing of the last segment of the message is complete. For the contents of key UEHB fields on entry and exit, see “Key UEHB Data Fields on Entry” on page 236 and “Key UEHB Data and Flag Fields on Exit” on page 238. See “User Exit Header Block (UEHB)” on page 247 for the UEHURSVD field and the entire UEHB.

Content of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents	
0	One of the following entry codes:	
	Entry Code	**Meaning**
	0	First (initial) entry to the exit routine for the message. A segment is always presented to the exit routine (and the UEH1SEG field in the UEHB is set) with this entry code. The buffer pointed to by the UEHCPYBF field contains the first segment of the message being processed. The flag in the UEHBFLG1 field indicates what kind of message it is.

4	Subsequent entry to the exit routine for the message. This entry code only applies to multisegment messages with three or more segments. The segment presented with this entry code is neither the first nor the last segment.
8	Last (final) entry to the exit routine for the message. This entry code only applies if the exit routine returned a return code of 0, 4, or 20 the last time it was called for this message (indicating that IMS continues to present the remaining segments to the exit routine).
12	Entry to exit routine after it requests storage. IMS passes the address of the buffer in the UEHUBUFF field. If storage is not available, UEHUBUFF contains 0 and the UEH1NSTG flag in the UEHBFLG1 field is set. The exit routine can attempt to get storage a second time, but if the second attempt is also unsuccessful, one of the following occurs: • If IMS is processing a command, it aborts further processing of this command. • If IMS is processing a system message, it examines the size of the requested storage. If the size requested is greater than twice the value of UEHCPYSZ (the size of the current segment plus 20 bytes), IMS terminates the exit routine for that message. If the size is equal to or less than this value, the exit routine waits for the storage to become available.
16	No message is presented to the exit routine. IMS aborted command processing because of errors in the command. IMS issues error messages, termination messages, or both. Command responses that have been built and passed to the exit routine are canceled. A new response is built if the error is encountered while a `/DISPLAY` command response is being built.

1	Address of the UEHB. (See 236 and 238for key entry and exit fields. See 247 for the meaning and location of all UEHB fields.)
7	Address of the communication terminal block (CTB).
9	Address of the communication line block (CLB) or partition specification table (PST).
11	Address of the system contents directory (SCD).
13	Address of the save area. The exit routine must not change the first three words. For external requests, the exit routine can chain down one save area to obtain the next available save area.
14	Return address to IMS.
15	Entry point of exit routine.

Key UEHB Data Fields on Entry

The following are key UEHB entry fields. For the meaning and location of these and other fields in the UEHB, see 247.

UEHCPYBF Address of the copy of the message IMS passed to the exit routine. If the UEH1SEG flag is set, the buffer contains a pointer to a copy of the message. If the UEH1CPYP flag in the UEHBFLG1 field is set, the buffer contains a copy of the first segment of a system message. If the UEH1CMD flag is set, the buffer contains a copy of the first segment of a command. If the UEH1CMD flag is set and the entry code is non-0, this field contains a copy of a segment of a command response.

UEHECMD Address of the edited command buffer if this is the final entry to the

	exit routine and the UEH1ECMD flag in the UEHBFLG1 field was set (requesting the edited command buffer) the first time IMS called the exit routine.
UEHUBUFF	Address of an additional storage buffer, if the exit routine requests storage. If additional storage was not available, this field contains 0, and the UEH1NSTG flag in the UEHBFLG1 field is set.

Content of Registers on Exit

Before returning to IMS, the exit routine must restore all registers except for registers 0, 1, and 15, which contain the following:

Register	Contents
0	If register 15 contains a return code of 0 or 8 (and your exit routine sets the destination for the first time or changes it), this register contains the address of the alternate destination name. The alternate destination can be a transaction or an LTERM. The alternate destination name must be 8 bytes, left-justified with blanks. If the alternate destination is not a valid transaction or LTERM and the Extended Terminal Option (ETO) is set, a dynamic LTERM is created. (For more information on the ETO feature, see the *IMS/ESA Administration Guide: Transaction Manager*.) If register 15 contains a return code of 16 (requesting additional storage), this register contains the size of storage requested for the user buffer. If the alternate destination was set with a previous return code of 0 (and your exit routine does not change it), this register contains 0.
1	If register 15 contains a return code of 0 or 8, this register contains the address of the segment to insert to the alternate destination or register 1 contains 0 to enqueue a previously inserted segment. If the segment to be inserted is the final segment, register 1 must contain the address of the message segment. If the segment is longer than the original segment (such as when your exit routine changes a message), and the device associated with the LTERM does not support the segment length, the terminal device can truncate the segment.
15	One of the following return codes:

Return Code	Meaning
0	Insert the segment to the alternate destination and continue presenting the remainder of the segments to the exit routine.
4	Do not insert the segment to an alternate destination. The exit routine can change the segment, or it can set the segment length to 0 to delete the segment. IMS continues to present remaining segments to the exit routine, which it can also change or delete. If register 0 contains 8 on entry, this return code causes all previously inserted segments to be enqueued.
8	If register 1 contains 0, this return code instructs IMS to enqueue the **previously** inserted segments to the alternate destination (not to insert new segments). Processing is considered complete. If register 1 contains the address of the segment to insert, this return code instructs IMS to insert the current segment, if there is one, to the alternate destination and enqueue all inserted segments. Even if there are additional segments, this return code indicates that the remaining segments are **not** to be presented to the exit routine. Processing is considered complete.
12	Cancel any segment already inserted to the alternate destination and indicate processing is complete.
16	Cancel any segment already inserted to the alternate destination and indicate processing is complete.
20	Cancel all prior segments inserted to an alternate destination and continue presenting the remainder of the segments to the exit routine.

IMS checks to make sure that the return codes and alternate destination name are valid. If an invalid return code or an invalid alternate destination is returned, the exit routine is disabled for the remainder of the segments and is not called. IMS sends a trace record and a DFS2180I AUTOMATED OPERATOR USER EXIT ERROR - CODE=x message to the master terminal.

Key UEHB Data and Flag Fields on Exit

The following are key UEHB exit fields. For the meaning and location of these and other fields in the UEHB, see 247.

UEHCPYBF
Address of buffer that contains a copy of the segment going to the master terminal if a system message is being processed. If the exit routine changes the length of the segment in UEHCPYBF, the LL field must also be changed to reflect the new length. The LL field can be increased by up to 20 bytes or can be set to 0 (to delete the system message destined for the master terminal).

UEH1ECMD
Flag in the UEHBFLG1 field indicating that the exit routine requests the edited command buffer. The exit routine must set this flag the first time it is called.

UEHURSVD
The 20 bytes of storage reserved for the exit routine. The exit routine can use UEHURSVD to save the message being processed, entry codes, or flags between each invocation of the exit routine for a particular message.

Messages Inserted to Transactions by This Routine

When an AO application obtains a message that was inserted by the exit routine by issuing a GU call, the application I/O PCB contains an inputting LTERM name. IMS determines the LTERM name as follows:

- If IMS calls the exit routine because of a system message, the inputting LTERM name is the master terminal name.
- If IMS calls the exit routine because of command input, the inputting LTERM name is the LTERM that entered the command.

Restriction: In a shared queues environment, transactions that must be processed by the local IMS subsystem must be defined as SERIAL to process correctly. A transaction that is not defined as SERIAL can be processed on any IMS subsystem that has that transaction defined.

AO Functions and How to Implement Them

You can write this exit routine to perform a number of functions. The following sections describe how to use the functions supported by the exit routine as listed in "About This Routine" on page 227. You can use this example as a guideline for writing your own exit routine. This example shows how to perform each of the functions on all segments of a multisegment message. You can write the exit routine to support only single-segment messages. This example requests a user buffer for some of the functions in which to store a copy of the message segment. You can use something else in which to store a copy of the message segment. The functions and register contents are summarized in Table 44 on page 244 and Table 45 on page 245.

Ignore Selected Segments or an Entire Message

The exit routine is called for system messages destined for the master terminal, operator-entered commands, and command responses regardless of whether the exit routine is interested in the message. If the exit routine is not interested in the message segment, it can set the exit registers to ignore them and resume processing. The segment is ignored by setting the following register:

Register 0 on entry	Registers on exit
0	Register 15 = 12

Send Copy of Message to Alternate Destination

You can write the exit routine so that IMS sends a copy of a system message destined for the master terminal, an operator-entered command, or a command response to an alternate destination in addition to the original destination. For the first entry to the exit routine, insert the segment to the alternate destination and request remaining segments, if there are any, by setting the following register:

Register 0 on entry	Registers on exit
0	Register 0 = address of alternate destination name Register 1 = address of message (UEHCPYBF) Register 15 = 0

For subsequent entries that are not the last entry, insert the segment to the alternate destination and request remaining segments by setting the following register:

Register 0 on entry	**Registers on exit**
4	Register 0 = address of alternate destination name Register 1 = address of message (UEHCPYBF) Register 15 = 0

For the last entry to the exit routine, insert the segment to the alternate destination, enqueue all of the segments, and indicate that processing is complete by setting the following register:

Register 0 on entry	**Registers on exit**
8	Register 0 = address of alternate destination name Register 1 = address of message (UEHCPYBF) Register 15 = 8

Send New Message to Alternate Destination

You can write the exit routine to send a new message to an alternate destination for each system message, operator-entered command, or command response that is passed to the exit routine. The original message proceeds to its destination unchanged, and a completely new message is sent to an alternate destination.

For the first entry to the exit routine for this message, the exit routine must request the storage in which to build each segment of the new message. The buffer requested during this initial entry must be large enough to fit the largest message segment you plan to send. The exit routine cannot request additional storage during subsequent entries for this message. Request enough storage for a message segment by setting the following register:

Register 0 on entry	**Registers on exit**
0	Register 0 = size of message segment Register 15 = 16

For the next entry to the exit routine after successfully getting storage for this message segment, move the first segment of the new message to the user buffer (UEHUBUFF), and set the message length in the first 2 bytes. Insert the message segment to the alternate destination and request remaining segments, if any, by setting the following register:

Register 0 on entry	**Registers on exit**
12	Register 0 = address of alternate destination name Register 1 = address of message segment (UEHUBUFF) Register 15 = 0

For subsequent entries that are not the last entry, move the next segment of the new message into the user buffer. The user buffer is reused for each segment of the message. Set the message length in the first 2 bytes of the user buffer. Insert the message segment to the alternate destination and request the remaining message segments by setting the following register:

Register 0 on entry	Registers on exit
4	Register 0 = address of alternate destination name Register 1 = address of message segment (UEHUBUFF) Register 15 = 0

For the last entry to the exit routine for this message, move the last segment of the new message into the user buffer. Set the message length in the first 2 bytes. Insert the segment to the alternate destination, and indicate that processing is complete by setting the following register:

Register 0 on entry	Registers on exit
8	Register 0 = address of alternate destination name Register 1 = address of message segment (UEHUBUFF) Register 15 = 8

Change System Message Text

The original text of a system message destined for the master terminal can be changed. The system message that is passed to the exit routine includes 20 bytes that are added to the end of the message. The exit routine can use these 20 bytes. Changes to the system message are limited to the original message length, plus 20 bytes. If the changed message includes the 20-byte area provided at the end, the exit routine must increment the message length field by 20. (The exit routine cannot change original command and command response messages, but it can change a copy of these messages. For more information, see “Change Message Text and Send to Alternate Destination”.)

For each entry to the exit routine for this message, change the system message text. For the first entry to the exit routine, allow the changed segment to proceed to its master terminal destination and request remaining message segments by setting the following register:

Register 0 on entry	Registers on exit
0 or 8	Register 15 = 4

Change Message Text and Send to Alternate Destination

The exit routine can change the copy of a system message destined for the master terminal and send the changed message to both the master terminal and an alternate destination. (If the copies sent to the master terminal and the alternate destination are different, your exit routine needs to request storage for the user buffer for the copy sent to the alternate destination.) The exit routine cannot change the copy of the command or command response that is in the copy buffer.

The copy of the message passed to the exit routine has an additional 20 bytes added to the end, which the exit routine can use. Changes to the message are limited to the original message length, plus this 20 bytes. If the changed message includes the 20-byte area, the exit routine must increment the message length field by 20.

For the first entry to the exit routine for this message, change the message text. If the changed message includes the 20-byte area provided at the end, increment the message length field by 20. Insert the segment to an alternate destination, and request the remaining segments by setting the following register:

Register 0 on entry	**Registers on exit**
0	Register 0 = address of the alternate destination name Register 1 = address of the message (UEHCPYBF) Register 15 = 0

For subsequent entries that are not the last entry, change the message segment text. If the changed message includes the 20-byte area provided at the end, increment the message length field by 20. Insert the message segment, and send it to an alternate destination by setting the following register:

Register 0 on entry	**Registers on exit**
4	Register 0 = address of the alternate destination name Register 1 = address of the message (UEHCPYBF) Register 15 = 0

For the last entry to the exit routine for this message, change the message text. If the changed message includes the 20-byte area provided at the end, increment the message length field by 20. Insert the segment to an alternate destination, enqueue all of the segments, and indicate that processing is complete by setting the following register:

Register 0 on entry	**Registers on exit**
8	Register 0 = address of the alternate destination name Register 1 = address of the message (UEHCPYBF) Register 15 = 8

Delete System Message to MTO

Your exit routine can delete a system message segment that is destined for the master terminal. (It cannot delete commands and command responses.) Delete a segment by setting the length field in the message buffer (the first two bytes) to 0.

For the first entry to the exit routine for the message, set the length field of the message to 0, and obtain the second segment by setting the following register:

Register 0 on entry	**Registers on exit**
0	Register 15 = 4

For subsequent entries that are not the last entry, set the length field of the message to 0, and obtain the next segment by setting the following register:

Register 0 on entry	**Registers on exit**
4	Register 15 = 4

For the final entry to the exit routine for the message, set the length field of the message to 0, and indicate that processing is complete by setting the following register:

Register 0 on entry	Registers on exit
8	Register 15 = 12

Delete System Message to MTO and Send Copy to Alternate Destination

Your exit routine can delete a system message destined for the master terminal and send a copy to an alternate destination instead. (It cannot delete commands and command responses.) Before deleting the system message, the exit routine must request storage for a user buffer in which to put a second copy of the message. Your exit routine must request enough storage to fit the largest segment of the message.

For the first entry to the exit routine for the system message, your exit routine can request storage by setting the following register:

Register 0 on entry	Registers on exit
0	Register 0 = size of the largest message segment Register 15 = 16

For the next entry after successfully getting storage for the largest message segment, move the first segment of the message from UEHCPYBF into the user buffer (UEHUBUFF), including the length in the first 2 bytes. Delete the message destined for the master terminal by setting the length field of the message segment pointed to by UEHCPYBF to 0. Insert the message copy to the alternate destination, and request the next segment by setting the following register on exit:

Register 0 on entry	Registers on exit
12	Register 0 = address of the alternate destination name Register 1 = address of message segment (UEHUBUFF) Register 15 = 0

For the last entry to the exit routine for this message, move the last segment of the message into the user buffer. The user buffer is reused for each segment of the message. Delete the last segment destined for the master terminal by setting the length field of the message segment pointed to by UEHCPYBF to 0. Insert the last segment, enqueue the entire message, and indicate that processing is complete by setting the following register on exit:

Register 0 on entry	Registers on exit
8	Register 0 = address of alternate destination name Register 1 = address of the message segment (UEHUBUFF) Register 15 = 8

Request the Edited Command Buffer

Your exit routine can request the edited buffer that was created for an input command. (For the format of the command buffer, see Figure 15 on page 234.)

On first entry, request the edited command buffer by setting flag UEH1ECMD on in the UEHBFLG1 field. Request the next command response segment by setting the following register on exit:

Register 0 on entry	**Registers on exit**
0	Register 15 = 4

For subsequent entries that are not the last entry to the exit routine for this command response, continue requesting the next command response segment by setting the following register on exit:

Register 0 on entry	**Registers on exit**
4	Register 15 = 4

For the last entry to the exit routine for this command response message, the UEHECMD field contains the address of the edited command buffer. If the edited command buffer is not available (such as when there are command syntax errors), the UEH1CBNA flag is set in the UEHBFLG1 field, and the UEHECMD field contains 0.

Setting Up the Exit Registers

The following figures describe how to set up exit registers to perform certain functions for single- and multisegment messages. (Each function is described in greater detail in “AO Functions and How to Implement Them” on page 239.) Refer to both figures if you are writing your exit routine to support single- and multisegment messages. If you can identify which messages are single-segment messages and which are multisegment messages, you can write the exit routine to handle each type differently. (For more information on these messages, see “Single- and Multisegment Messages” on page 232.)

Single Segment Messages

Table 44 shows how to set up registers on exit for single-segment messages. If your exit routine only examines single-segment messages, or if you can identify which messages are single segment messages (and can use this logic), you can use the information in this figure to write your exit routine.

Table 44. Exit Functions for Single Segment Messages

Function	Register 0 on entry	UEHCPYBF Length Field on exit	Register 0 on exit	Register 1 on exit	Register 15 on exit
Ignore entire message	0				12
Send copy of message segment to alternate destination	0		Address of alternate destination name	Address of message (UEHCPYBF)	8
Send new message to alternate destination	0		Size of message		16
	12		Address of alternate destination name	Address of message (UEHUBUFF)	8
Change system message	0	Length + 20			8

Table 44. Exit Functions for Single Segment Messages (continued)

Function	Register 0 on entry	UEHCPYBF Length Field on exit	Register 0 on exit	Register 1 on exit	Register 15 on exit
Change message segment and send to alternate destination	0	Length + 20	Address of alternate destination name	Address of message (UEHCPYBF)	8
Delete system message to master terminal	0	0			8
Delete system message to master terminal and send copy to alternate destination	0		Size of message		16
	12	0	Address of alternate destination name	Address of message (UEHUBUFF)	8

Multisegment Messages

Table 45 shows how to set up the registers on exit for multisegment messages. If your exit routine examines multisegment messages, or if you can identify which messages are multisegment messages (and can use this logic), you can use the information in this figure to write your exit routine.

Table 45. Exit Functions for Multisegment Messages

Function	Register 0 on entry	UEHCPYBF Length Field on exit	Register 0 on exit	Register 1 on exit	Register 15 on exit
Ignore entire message	0				12
Send copy of message to alternate destination for each segment	0		Address of alternate destination name	Address of message (UEHCPYBF)	0
	4		Address of alternate destination name	Address of message (UEHCPYBF)	0
	8		Address of alternate destination name	Address of message (UEHCPYBF)	8

Table 45. Exit Functions for Multisegment Messages (continued)

Function	Register 0 on entry	UEHCPYBF Length Field on exit	Register 0 on exit	Register 1 on exit	Register 15 on exit
Send new message to alternate destination for each segment	0		Size of storage to get for largest message segment		16
	12		Address of alternate destination name	Address of message segment (UEHUBUFF) with length field set	0
	4		Address of alternate destination name	Address of message segment (UEHUBUFF) with length field set	0
	8		Address of alternate destination name	Address of message segment (UEHUBUFF) with length field set	8
Change each segment of system message	0	Length + 20			4
	8	Length + 20			4
Change each segment of a message and send to alternate destination	0	Length + 20	Address of alternate destination name	Address of message (UEHCPYBF)	0
	4	Length + 20	Address of alternate destination name	Address of message (UEHCPYBF)	0
	8	Length + 20	Address of alternate destination name	Address of message (UEHCPYBF)	8
Delete each segment of system message to master terminal	0	0			4
	8	0			12

Table 45. Exit Functions for Multisegment Messages (continued)

Function	Register 0 on entry	UEHCPYBF Length Field on exit	Register 0 on exit	Register 1 on exit	Register 15 on exit
Delete each system segment to master terminal and send copy to alternate destination	0		Size of storage to get for largest message segment		16
	12	0	Address of alternate destination name	Address of message segment (UEHUBUFF)	0
	8	0	Address of alternate destination name	Address of message segment (UEHUBUFF) with length field set	8
Request the Edited Command Buffer	0				4
	4				4
	8				12

User Exit Header Block (UEHB)

The UEHB contains the following data and flag fields. Table 46 indicates the field name, length in bytes, and description of the data fields, and it indicates the field name, hexadecimal value, and meaning of the flag fields. (For more high-level information on the UEHB's use, see "Communicating with IMS" on page 235.)

Data and flag fields in the UEHB can be grouped into one of three categories, depending on how the exit routine can use them.

- Modifiable: The exit routine can change these fields to communicate with IMS or to use as a work field.
- Read only: The exit routine can read but not modify these fields.
- Reserved: The exit routine cannot use these fields. They are reserved for use by IMS.

Table 46. UEHB Field Descriptions

Field	Length/Value	Description
UEHSRCE	4 bytes	Address of source CNT. Usage = read only. This field points to the source LTERM of the message segment. For a system message, the source is the master LTERM. For a command, the source is the LTERM where the command was entered.
UEHDEST	4 bytes	Address of destination CNT. Usage = read only. This field points to the destination of message segment. This is the destination of the 'presented' message segment, not the alternate destination that the exit routine can define.

Table 46. UEHB Field Descriptions (continued)

Field	Length/Value	Description
UEHUBUFF	4 bytes.	Address of user buffer. Usage = read only. The buffer pointed to by this field is acquired when the exit routine returns a return code of 16 in register 15. The buffer can contain a copy of a message to be inserted to an alternate destination. If the buffer contains a copy of a message, the exit routine must update the 2-byte length field. If there is no message in the buffer, the length field is not necessary.
UEHCPYBF	4 bytes	Address of exit routine copy buffer. Usage = read only. The buffer pointed to by this field contains a copy of the system message segment, command, or command response that IMS passes to the exit routine. This area is the size of the message segment + 20 bytes (for modification). The first two bytes are the length field. For each entry to the exit routine, IMS reuses the copy buffer if the buffer created for a prior call is greater than or equal to the size that the current call requires. Otherwise, IMS frees the prior buffer and creates a new one.
UEHECMD	4 bytes	Address of edited command buffer. Usage = read only. On last entry to the exit routine, this field contains the address of the edited command buffer if the UEH1ECMD flag in the UEHBFLG1 field was set during the first entry to the exit routine for the message.
UEHIPCB	4 bytes	Address of input PCB. Usage = reserved.
UEHIWRK1	4 bytes	Internal work area. Usage = reserved.
UEHIWRK2	4 bytes	Internal work area. Usage = reserved.
UEHIWRK3	4 bytes	Internal work area. Usage = reserved.
UEHBMODN	8 bytes	MFS MOD name. Usage = modifiable. The MFS MOD name sent with the message to the alternate destination that the exit routine specifies.
UEHPOPCB	28 bytes	Alternate PCB. Usage = reserved. Used for either AO transaction or any other transaction or LTERM destination that the AO specifies.
UEHSCPCB	28 bytes	Secondary master PCB. Usage = reserved.
UEHCPYSZ	2 bytes	Size of UEHCPYBF copy buffer. Usage = reserved.

Table 46. UEHB Field Descriptions (continued)

Field	Length/Value	Description
UEHLINE	4 bytes	Line number. Usage = read only. BTAM line number (in binary) or 0. If a command was entered, this is the line number of the BTAM terminal that entered the command. If a system message is being passed and the master terminal is a BTAM line, this is the line number of the BTAM master terminal.
UEHTERM	4 bytes	PTERM number. Usage = read only. BTAM PTERM number (in binary) or 0. If a command was entered, this is the PTERM number of the BTAM terminal that entered the command. If a system message is being passed and the master terminal is a BTAM line, this is the PTERM number of the BTAM master terminal.
UEHNODE	8 bytes	Nodename. Usage = read only. VTAM nodename or 0. This field references the same 8 bytes of storage as the UEHLINE and UEHTERM fields. If a command was entered, this is the nodename of the VTAM terminal that entered the command. If a system message is being passed and the master terminal is a VTAM node, this is the nodename of the VTAM master terminal node.
UEHHSQN	8 bytes	User name. Usage = read only. User name of user signed on to node or the Intersystem Communication (ISC) user associated with the node that entered the command, if UEH1CMD is set on in UEHBFLG1. User name or 0.
UEHOCALL	2 bytes	Usage = reserved.
UEHBFLG1	1 byte	Flag byte 1 for AOI and exit routine as follows:
UEH1ECMD	X'80'	Indicates that the exit routine requests the edited command buffer. Usage = modifiable. If the exit routine sets this flag on the first entry, the UEHECMD field points to the edited command buffer on the last entry. Also see UEH1CBNA flag.
UEH1SEG	X'40'	Indicates that a segment is presented to the exit routine. Usage = read only. This flag is set when a segment is present in the UEHCPYBF field for the exit routine's examination. It is reset if entry code = 8 and prior call was PUT MOVE (segment already presented to exit routine on previous call with entry code = 0 or 4).

Table 46. UEHB Field Descriptions (continued)

Field	Length/Value	Description
UEH1CPYP	X'20'	Indicates that the exit routine was called for a system message destined for the master terminal. Usage = read only. This flag is set when an asynchronous system message caused IMS to call the exit routine. A UEHB is created and the message is present in the UEHCPYBF field. The copy buffer contains the message plus 20 bytes (which the exit routine can modify).
UEH1CMD	X'10'	Indicates that the exit routine was called for a command or command response (if the entry code is non-0). Usage = read only. This flag is set when a command caused IMS to call the exit routine. A UEHB is created and the command is present in the UEHCPYBF field. This flag is set until command processing is completed.
UEH1NSTG	X'08'	Storage not available for the user buffer. Usage = read only. Set when a conditional request for storage for the user buffer cannot be satisfied.
UEH1CBNA	X'04'	Edited command buffer not available Usage = read only. Set if the edited command buffer was not constructed by the command processor due to errors.
UEH1PSTD	X'02'	PST dispatch. Usage = reserved. System message being issued as a result of application program processing.
UEHBFLG2	1 byte	Flag byte 2 for AOI.
UEH2BYP	X'80'	Exit routine does not want rest of message. Usage = reserved. Set when the exit routine returns a return code of 8 or 12 indicating no more message segments are to be presented.
UEH2POTR	X'40'	Alternate destination found. Usage = reserved. Set after the alternate destination is successfully found, when register 15 contains a 0 return code and an 8-byte alternate destination name is in register 0. The alternate destination may be a transaction or LTERM.

UEH2ILOC	X'10'	INSERT LOCATE was last call.
		Usage= reserved.
		Indicates that the prior segment is to be presented to the exit routine.

UEH2UENT	X'08'	Exit routine was entered at least once. Usage = reserved. Used to determine if entry code 0 is to be set.
UEH2LAST	X'04'	Set when entry code 8 is set upon entry. Usage = reserved. Used to indicate that the exit routine was entered for the last time.

Table 46. UEHB Field Descriptions (continued)

Field	Length/Value	Description
UEH2NCUR	X'02'	Set when current call is not yet processed. Usage = reserved. Used when entered for a final call but the exit routine had not yet been entered for the first time.
UEH2QNOP	X'01'	Set when call is not to be passed to the queue manager (QMGR). Usage = reserved. Used to indicate that a PUT MOVE should not be done if the exit routine is deleting a system message to the primary master terminal.
UEHBFLG3	1 byte	Flag byte 3 for AOI.
UEH3ILOC	X'80'	Current call is INSERT LOCATE. Usage = reserved.
UEH3PUTM	X'40'	Current call is PUT MOVE. Usage = reserved.
UEH3CANO	X'20'	Current call is CANCEL OUTPUT. Usage = reserved.
UEH3ENQ	X'10'	Current call is ENQUEUE. Usage = reserved.
UEH3TERM	X'08'	Current call is AOI TERMINATION. Usage = reserved.
UEH3VSEG	X'04'	Segment exists for M/T. Usage = reserved.
UEHBFLG4	1 byte	Error flag byte.
UEH4ERRM	X'80'	AOI error message in progress. Usage = reserved.
UEH4SMER	X'40'	Secondary master terminal error. Usage = reserved.
UEH4FAIL	X'20'	Current call will be failed. Usage = reserved.
UEH4UEHB	X'10'	Previous UEHB exists. Usage = reserved.
UEHBERRC	1 byte	QAOI error code as follows:
UEHBERR1	C'1'	Invalid alternate destination. Usage = reserved.
UEHBERR2	C'2'	Queue manager return code. Usage = reserved.
UEHBERR3	C'3'	Invalid exit routine return code. Usage = reserved.
UEHBERR4	C'4'	Multiple user buffer request. Usage = reserved.
UEHBERR5	C'5'	User buffer storage not available. Usage = reserved.
UEHBERR6	C'6'	Previous UEHB exists. Usage = reserved.
UEHBERR7	C'7'	Usage= reserved.
UEHBFLG5	1 byte	Flag byte 5 for AOI.

Table 46. UEHB Field Descriptions (continued)

Field	Length/Value	Description
UEH5LTRM	X'80'	Usage = reserved. Dynamic LTERM marked in use.
UEHIRSVD	2 bytes	Usage = reserved.
UEHURSVD	20 bytes	Work area reserved for exit routine. Usage = modifiable. The exit routine can use this field to keep track of message numbers, entry codes, and flags between invocations of the exit routine for a particular message. UEHURSVD can be used to tie segments of a multisegment message together, since remaining segments do not contain the message number.

Chapter 30. Build Security Environment Exit Routine (DFSBSEX0)

The Build Security Environment exit routine provides users with a mechanism to tell IMS whether or not to build the RACF or equivalent security environment in an IMS dependent region for an application that has received its input message from neither OTMA nor an LU 6.2 device.

For example, you might want to use this exit routine to tell IMS to build the security environment for an IMS modified application (that is, the application issues an APPC/MVS outbound allocate) that you know does not receive its input from an LU 6.2 device.

In this Chapter:

You can also use this exit routine to request that IMS bypass some part of the security processing in the dependent region when one of the following occurs for a message that did not originate from an OTMA or LU6.2 device:

- CHNG call.
- AUTH call.
- Deferred conversational program switch on the local system when the system where the inputting terminal is active. Security authorization for the referred conversational program switch occurs only on the local system.

About this Routine

The Build Security Environment exit routine receives control before the first or next input message is given to an IMS application program and the input message is from neither OTMA nor an LU 6.2 device.

This routine executes in key 7, non-cross-memory mode under the dependent region TCB.

Table 47 shows the attributes of the Build Security Environment exit routine.

Table 47. Build Security Environment Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSBSEX0.

Link editing	Follow the guidelines described in "Link-Editing the Routines" on page 5.
	
	You must write this routine using reentrant coding techniques. You must link your routine into the IMS.RESLIB library.
	
	If you choose to use IMS Callable Services, you must link DFSCSI00 with your routine. The following is an example of the link-edit JCL statements needed:
	
	`INCLUDE LOAD(DFSBSEX0)`
	`INCLUDE LOAD(DFSCSI00)`
	`ENTRY   DFSBSEX0`
	`NAME    DFSBSEX0(R)`

Table 47. Build Security Environment Exit Routine Attributes (continued)

Attribute	Description
Including the routine	No special steps are required to include this routine.
IMS callable services	In order to use IMS Callable Services with this routine, you must determine if IMS passed a Callable Services token to you by examining field SXPLATOK in the IMS Standard User Exit Parameter List. See “IMS Standard User Exit Parameter List (SXPL)” on page 255 for more information on the IMS Standard User Exit Parameter List for this routine. If SXPLATOK is zero, you cannot use IMS Callable Services with this routine. If SXPLATOK is non-zero, the only IMS Callable Service you may use is storage services. You do not have to invoke DFSCSII0 to initialize IMS Callable Services. You can use the 256-byte work area pointed to by SXPLAWRK in the IMS Standard User Exit Parameter List for the required parameter lists for your call to DFSCSIF0. See “Using IMS Callable Services” on page 8 for more information.
Sample routine location	No sample exit routine is provided.

Communicating with IMS

IMS uses the entry registers, the Standard User exit parameter list (SXPL), and the Build Security Environment exit (BSE) parameter list to communicate with this routine.

This routine uses register 15 to communicate with IMS.

Contents of Registers on Entry

Register	Contents
1	Address of the IMS Standard User exit parameter list (SXPL). See “IMS Standard User Exit Parameter List (SXPL)” on page 255.
13	Address of a single standard MVS save area.
14	Return address to IMS.
15	Address of DFSBSEX0.

All other registers are undefined.

Contents of Registers on Exit

Register	Contents
15	Return code indicating requested action:

00	IMS is not to build the security environment during the scheduling phase of the transaction. The security environment can be built later if needed for processing a CHNG call, AUTH call, or a deferred conversational program switch.
04	IMS is to build the security environment during the scheduling phase of the transaction. If the security environment is needed later by a CHNG call, AUTH call, or a deferred conversational program switch, this same security environment is used. If the application program does not ever need the security environment, the build of the security environment is unnecessary.
08	Invoke the SAF interface (RACF, or equivalent product) on a CHNG call, an AUTH call, and a deferred conversational program switch, but bypass the dynamic creation of the security environment. If the transaction is running in the local system, and the user who entered the transaction is still signed on, the security environment created by SIGNON is used. Otherwise, the default security environment of the IMS control region or the IMS dependent region is used for the SAF call. The processing for this return call is running in cross memory mode. It is not a BMP or is not running with LSO=Y. The processing uses the security environment of the dependent region. In IMS 5.1, the security environment of the Control Region was always used as the default.
12	Bypass invoking the SAF interface on a CHNG call, an AUTH call, and a deferred conversational program switch.
16	Bypass invoking the SAF interface on a CHNG call, an AUTH call, and a deferred conversational program switch, and bypass the calls to the DFSCTRN0 and DFSCTSE0 user exits.

Notes:

1. For return codes 08, 12 and 16, IMS does not dynamically build the security environment during transaction scheduling, or later for a CHNG call, an AUTH call, or a deferred conversational program switch.
2. When return code 16 is used, the application gets a status code in the IOPCB of blanks. For the AUTH call, the status field in the I/O area has the value 24 (X'18'): transaction authorization not active.

All other registers are to be restored by this routine.

IMS Standard User Exit Parameter List (SXPL)

The address of this parameter list is contained in register 1 on entry to this routine. Table 48 defines the format of this parameter list as mapped by the macro DFSSXPL.

Table 48. IMS SXPL for DFSBSEX0

Offset	Field Name	Contents/Description
X'00'	SXPLVER	Address of a fullword containing the parameter list version number.
X'04'	SXPLATOK	0 or the address of a fullword containing the Callable Services token for this instance of the routine.
X'08'	SXPLAWRK	Address of a 256-byte work area.
X'0C'	SXPLFSPL	Address of the BSE parameter list.

Table 48. IMS SXPL for DFSBSEX0 (continued)

Offset	Field Name	Contents/Description
X'10'	SXPLINTX	Address of the user data table loaded by DSINTX0 at IMS initialization time. This field is valid only in IMS environments where DFSINTX0 is called. It will be X'80000000' in any other environment. For more information about the DFSINTX0 exit routine, see "Chapter 36. Initialization Exit Routine (DFSINTX0)" on page 289.

Build Security Environment Exit (BSE) Parameter List

The address of the BSE parameter list on entry to this routine is contained in field SXPLFSPL of the IMS Standard User Exit parameter list. Table 49 describes the BSE parameter list.

Table 49. BSE Parameter List

Offset	Field Length	Description
X'00'	4 bytes	Transaction scheduling class.
X'04'	8 bytes	Transaction code of the input transaction.
X'0C'	8 bytes	PSB name.
X'14'	8 bytes	Program name.
X'1C'	8 bytes	User ID. Specifies one of the following: • Actual user ID of the user who entered the transaction. • LTERM name of the terminal from which the transaction was entered. • Blanks. This is the user ID for which the security environment will be built if requested by this exit routine.
X'24'	8 bytes	Group name.
X'2C'	32 bytes	Application parameter (APARM= on dependent region JCL).
X'4C'	64 bytes	First 64 bytes of the input message or zeros if the input transaction is conversational.

Chapter 31. Conversational Abnormal Termination Exit Routine (DFSCONE0)

A conversational process terminates abnormally when:

- A conversation is ended by an `/EXIT` or `/START` command.
- A conversational application program terminates abnormally during a conversation.
- A conversational program fails to insert a message into a response PCB or into an alternate PCB that represents another conversational program.
- A noncorrectable IMS conversational error occurs.

If used, the Conversational Abnormal Termination exit routine can be scheduled twice: once when an `/EXIT` or `/START` command is issued, and again when an application program inserts a SPA, or when the conversational response is received from a remote system.

In this Chapter:

About This Routine

You can provide an application program to clean up, if required, when a conversation is prematurely terminated. Upon entry, this program’s I/O PCB contains the name of the terminal that had its conversation abended. An exit routine to schedule the application program is required. IMS provides an exit routine named DFSCONE0, or you can write your own. To use the IMS-provided routine, you must:

- Define a transaction code named DFSCONE.
- Write a nonconversational application program to be activated by DFSCONE.

When the exit routine (DFSCONE0) is finished, the IMS conversational processor determines whether the transaction DFSCONE has been defined. If DFSCONE is not defined, conversation terminates and the SPA is discarded. If DFSCONE is defined, the conversational processor schedules the transaction DFSCONE with the SPA of the terminated conversation as a nonconversational single-segment message.

As an alternative to the above, you can provide a more tailored exit routine. For example, you might want to interrogate the CCB to determine which transaction was in process when the conversation terminated, or you might want to inspect the SPA to find out what had occurred before the conversation terminated. No DL/I calls can be issued by your exit routine. A message processing program should be scheduled to handle database inquiries and updates or extensive analysis of the conversation. The application program can send messages to the terminal associated with the terminated conversation.

To cause your application program to be scheduled, your exit routine must:

- Place the 8-byte name of the nonconversational transaction into the SPA (offset 6 bytes into the SPA).
- Set the desired length of the SPA.
- Insert information to be communicated to the scheduled program into the SPA.

- Set a return code of X'10' in register 15.

The transaction code inserted into the SPA must be for a valid, nonconversational transaction. Otherwise, no transaction will be scheduled, the SPA is discarded, and the response message (if available) is sent to the input terminal.

Attributes of the Routine

Table 50 shows the attributes for the Conversational Abnormal Termination exit routine.

Table 50. Conversational Abnormal Termination Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSCONE0.
Link editing	Follow the guidelines described in “Link-Editing the Routines” on page 5. This routine is automatically link to DFSCSI00 by IMS. No additional linking is required to use IMS Callable Services.
Including the routine	To include a user-written exit routine, you must replace the default DFSCONE0 in IMS.LOAD with your own DFSCONE0 before link-editing the IMS nucleus and the composite load module DFSRST00. To use the default DFSCONE0, you need only define the transaction DFSCONE.
IMS callable services	To use IMS Callable Services with this routine, you must issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function specific parameter list for the desired callable service. For details on using the initialization call, see “Step 2. Initializing IMS Callable Services (DFSCSII0)” on page 11. Use the ECB in Register 9 for IMS Callable Services.
Sample routine location	IMS.TMSOURCE (member name DFSCONE0).

Restrictions

In a shared queues environment IMS cannot pass a conversational SPA to DFSCONE0 after the input message is queued and before the output message is received. If an `/EXIT` command or a `/START NODE`, `LINE`, or `USER` command is entered before the output message is received, the conversation terminates, but the SPA is not passed to the exit routine.

If the conversation terminates in this manner, IMS calls the exit routine a second time when the output message is received and passes the most current version of the SPA to the exit routine.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of Registers on Entry

The contents of **register 0** identifies the cause of conversation termination as follows:

Byte	**Contents**
0	

Return Code	**Meaning**
00	`/EXIT` command was issued to the application program to terminate the conversation. Register 0 byte 3 provides further clarification.
01	The `/EXIT` command was issued by the terminal in conversation, causing the conversation to be terminated. No pointer to the CCB or CTB exists.
02	The `/EXIT` or `/START` command was issued by a different terminal than the one in conversation; this causes the conversation to be terminated.
04	The input CNT could not be found. The master terminal of the current system is set as the input terminal.
08	The transaction was discarded by the `/EXIT` command.

Byte	**Contents**
1	

Return Code	**Meaning**
01	Conversation was terminated previously by an /EXIT, /START, or IMS cold start. The conversation transaction processed successfully, and IMS is sending (queuing) the response message to the input terminal. See the return code meanings in "Contents of Registers on Exit" on page 261 for options you can use to process the response message and scratch pad (SPA).

Byte	**Contents**
2	Pointer to a parameter list that contains SPA processing options. See Table 51 on page 260 for the contents of this parameter list.

3 Vector describing the calling reason:

Vector	Reason
00	Conversational application program abended.
04	Reserved.
08	/EXIT command for input or other (remote) terminal processed.
0C	/START LINE or NODE command processed for terminal in conversation. The /START LINE command is valid only if no PTERMs are specified.
10	SPA received for an inactive conversation.
14	Inconsistent conversational definitions found in a multisystem conversation. Execute the /MSVERIFY command to show the inconsistencies.
18	/EXIT command terminated the conversation and the latest SPA is not currently available. (It is queued for processing in this system, or it is in the MSC network.) The SPA passed to the exit routine is the one from the previous step of the conversation. The exit routine is called with vector 10 when the current step in progress completes; at this time the latest (and last) SPA for the conversation is passed to the exit routine. This may not occur if an IMS restart results in the loss of the SPA in this or another IMS system.
1C	The explanation for the /START LINE or NODE command is the same as for 18 above.
20	A conversational application program terminated without inserting to a response PCB or an alternate PCB that represents another conversational program.
28	/EXIT command for input or other (remote) ISC terminal processed.
30	The MSC Link Receive Routing exit routine (DFSCMLR0) canceled the input transaction.

The contents of the remaining registers are:

Register	Contents
1	Address of the SPA.
6	Address of the CCB for the terminal in conversation, if the conversation is still active. Zero if the conversation is already terminated.
7	If zero, the conversation is already terminated. If positive, the register contains the address of the CTB for the terminal in conversation (if the conversation is active). If negative, the register contains the complemented address of the SPQB for the signed-off user, which may be the result of the exit being called because of an /EXIT CONV USER command.
09	Address of the ECB.
11	Address of the SCD.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of DFSCONE0.

Table 51. SPA Options Parameter List

Field	Description
CONESPAH	Maximum SPA length
CONESPAL	Current SPA length

Table 51. SPA Options Parameter List (continued)

Field	Description
CONEFLG1	Flag 1. This flag can be set as follows: **CONE1TDO (X'80')** If this flag is set, register 1 points to a SPA buffer that contains the SPA at the maximum length. If this flag is not set, register 1 points to a SPA that is the length of the SPA for the current transaction. Truncated data option is set for the SPA parameter in the TRANSACT macro. **CONE1SQ (X'40')** Shared queues are active.

Contents of Registers on Exit

Upon return to IMS, all registers must be restored except for register 15, which must contain one of the following return codes:

Reason Code	Meaning
00	Exit has completed all cleanup required; no further action is necessary. IMS does the following: • Terminates the conversation (if still active). • Discards the SPA. • Discards the response message (if available).
04	The conversation is ended. The name field is used as a transaction code for a new non-conversational transaction. The remaining data in the SPA is used as input data for a new transaction. IMS does the following: • Terminates the conversation (if still active). • Attempts to queue the SPA to the indicated transaction and schedule it. See note 1 on page 262. • Discards the response message (if available). See note 1 on page 262.
08	Exit has completed all cleanup required. No further action is necessary. IMS does the following: • Terminates the conversation (if still active). • Discards the SPA. • Sends the response message to the input terminal (if available). See note 2 on page 262.
0C	The conversation is ended. The name field is used as a transaction code for a new non-conversational transaction. The remaining data in the SPA is used as input for a new transaction. IMS does the following: • Terminates the conversation (if still active). • Attempts to queue the SPA to the indicated transaction and schedule it. • Sends the response message to the input terminal (if available). See note 2 on page 262.

10	The conversation is ended. The name field is used as a transaction code for a new non-conversational transaction. The remaining data in the SPA is used as input data for a new transaction. IMS does the following: • Terminates the conversation (if still active). • Attempts to queue the SPA to the indicated transaction and schedule it. See note 3. • Discards the response message (if available). See note 3.

Notes for Contents of Registers on Exit:

1. If the SPA cannot be queued to the transaction because the transaction is not defined or defined incorrectly, the response message is still discarded.
2. Upon entry, if bit 7 in register 0, byte 1, is set on (R0='XX01XXXX'), the response message is available.
3. If the SPA cannot be queued to the transaction because the transaction is not defined or defined incorrectly, the response message is not discarded but is sent to the input terminal. Upon entry, if bit 7 in register 0, byte 1, is set on (R0='XX01XXXX'), the response message is available.

Chapter 32. Fast Path Input Edit/Routing Exit Routine (DBFHAGU0)

IMS systems with a very high transaction rate use Fast Path's Expedited Message Handling Facility (EMH). EMH is a performance option that speeds up message processing by imposing restrictions upon message lengths and segmentation. To use EMH, an edit/routing routine must receive control from the Input exit routine and determine the eligibility of an incoming message for Fast Path processing. The sample exit provides the minimum level of support required to use IMS Fast Path.

In this Chapter:

About This Routine

This Fast Path input edit/routing exit routine, DBFHAGU0, replaces DBFSUIX0 or your installation-equivalent exit used in IMS 3.1 and earlier releases of IMS. In these earlier releases, DBFSUIX0 edited Fast Path input in a queue buffer and then moved the data to a Fast Path EMH buffer if the message was to be scheduled by Fast Path. DBFHAGU0 allows the application program to examine, change, or edit the input message in the input area, although the application program cannot move the data out of the input location.

In IMS 4.1 and later releases, the EMH buffer is dynamically allocated and might not be present at entry. Therefore, DBFHAGU0 can receive the message in an EMH buffer or queue buffer, depending on the terminal type. The exit routine is not permitted to move the data our of the input location. If the message is in a queue buffer at entry, the Fast Path system moves it to an EMH buffer. In editing the input message, the application should not increase the length beyond a length that fits in any message buffer.

If an EMH buffer cannot be obtained, the following message is sent so to the input terminal:

```
DFS3971 Unable to process Fast Path due to EMH buffer shortage
```

Table 52 shows the attributes for the Fast Path Input Edit/Routing exit routine.

Table 52. Fast Path Input Edit/Routing Exit Routine Attributes

Attribute	Description
IMS environments	*What environments use this routine?*
Naming convention	You must name this exit routine DBFHAGU0. You cannot override this name on the FPCTRL sysgen macro.
Link editing	This exit routine must be reentrant if APPC/IMS support is active.
Including the routine	DBFHAGU0 is a separately linked module in the IMS.RESLIB. IMS automatically loads it during Fast Path initialization. If IMS cannot find DBFHAGU0, IMS terminates abnormally with ABENDU1011 and displays the following message: `DFS2730A UNABLE TO LOAD FP INPUT ROUTING EXIT: DBFHAGU0`

Table 52. Fast Path Input Edit/Routing Exit Routine Attributes (continued)

Attribute	Description
IMS callable services	To use IMS Callable Services with this routine, you must issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. Use the ECB found at offset X'0' of the Fast Path Input Edit/Routing Exit parameter list for the DFSCSII0 call. For details on using the initialization call, see "Step 2. Initializing IMS Callable Services (DFSCSII0)" on page 11. This exit routine is automatically linked to DFSCSI00 by IMS. No additional linking is required to use callable services.
Sample routine location	IMS.TMSOURCE (member name DBFHAGU0).

Restrictions

You must rewrite your Fast Path Input Edit/Routing exit routine for this release of IMS, based on the DBFHAGU0 sample (located in the IMS.TMSOURCE library) and the guidelines in this chapter.

The exit routine cannot move the data out of the input location.

The exit routine must not increase the length of the message beyond a length that fits in any message buffer.

Expanding the Routine

A transaction that is not Fast Path-exclusive can be directed to EMH processing by an expanded edit/routing routine, based on some condition or conditions beyond transaction code. For example, certain transactions can be routed to EMH if they originate at specified physical or logical terminals or if they reference the content of some portion of the message (for example, account number). The user-supplied DBFHAGU0 would have to develop appropriate routing codes based on such conditions.

Using the Routine with Shared EMH Queues

If your installation uses shared EMH queues, DBFHAGU0 can place messages on the shared queue structure for processing by any sharing IMS subsystem in the sysplex.

You can modify the exit routine to specify an application name for the application program used to process Fast Path input messages. If you do not specify an application name, Fast Path locates the transaction or routing code in the local IMS subsystem. Fast Path rejects the input message if it cannot locate the transaction or routing code.

You can also specify a sysplex processing code that determines how a message transaction or routing code is processed. The following sysplex routing options are available:

Local First	Specifies that the message is processed on the local subsystem if an IFP region is available. If no IFP region is available, the message is passed to the EMH queue structure. A program name specified in the exit routine for message processing overrides the transaction or routing code. Local First is the default.

Local Only Specifies that Fast Path does not place the message on the EMH queue structure. Fast Path input messages are processed on the local IMS subsystem.

Global Only Specifies that Fast Path places the input message on the EMH queue structure. The application program that processes the input message must be active on all sharing IMS subsystems. If the application is not active, Fast Path discards the input message and issues an error message. A program name specified in the exit routine for message processing overrides the transaction or routing code.

Recommendation: To avoid implicit priority for Local Only messages over Local First messages, process Local First and Local Only messages under separate program names. IMS places Local Only messages on the balancing group (BALG) queue and Local First messages on the shared EMH queue. When an IFP region becomes available, it checks the BALG queue for messages to process before it checks the shared EMH queue. This sequence gives implicit priority to Local Only messages that are processed in the same program.

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the exit routine.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
R1	Address of Standard Exit Parameter List (see Table 53).
R13	Save area address.
R14	Return address to IMS.
R15	Entry point address of exit routine.

Table 53. Standard Exit Parameter List (Version 1)

Offset	Length	Description
+0	4	Address of standard exit parameter list version number
+4	4	Reserved
+8	4	Reserved
+12	4	Address of Fast Path Input Edit/Routing Exit parameter list

Table 54 lists the fast path exit parameters.

Table 54. Fast Path Input Edit/Routing Exit Parameter List

Offset	Length	Description
+0	4	ECB address.
+4	4	SCD address.
+8	4	Input message.
+12	4	Address of routing code table entry if this is a Fast Path exclusive transaction, or zero.

Table 54. Fast Path Input Edit/Routing Exit Parameter List (continued)

Offset	Length	Description
+16	4	Eight-character work area to supply a routing code name.
+20	4	Address of ESCD.
+24	4	The length of the EMH Buffer for this application.
+28	4	Address of the DBFHAGU0 extended parameter list. This parameter list exists if shared EMH queues are used. Otherwise, the extended parameter list is 0.

Table 55 lists the Extended Parameter list parameters.

Table 55. DBFHAGU0 Extended Parameter List

Offset	Length	Description
+0	4	Address of the 8-byte PSB name
+4	4	Sysplex processing code **0** Local First (Default) **4** Local Only **8** Global Only
+8	4	Address of the Local PSB name table[1]
+12	4	Address of the Global PSB name table[1]
+16	4	System definition code **0** Transaction Defined in local system **4** Transaction not defined in local system
+20	4	Input message code **0** Fast Path exclusive transaction **4** Fast Path potential transaction

Notes:

1. The sample DSECT for the local program name table and the global program name table can be found in the DBFPGNT macro.

Contents of Registers on Exit

Upon return, all registers must be restored except for register 1 and 15, which must contain the following:

Register	Contents
1	Message number to send to inputting terminal.
15	One of the following return codes:

Return Code	Meaning
00	Schedule via Fast Path. Register 3 points to the RCTE to be used.
04	Schedule via Fast Path using transaction code as the routing code.
08	Schedule via Fast Path using the routing code you provide.
12	Return to IMS for processing.
16	Schedule via Fast Path using transaction code if the routing code equal to transaction code is active; otherwise, let IMS process it.
20	Schedule via Fast Path using routing code provided the routing code is active; otherwise, let IMS process it. This is the same action as user exit return code 08.
24	Discard input, send message from user table back to inputting terminal.
28	Discard input, send message from system message table.

Chapter 33. Front-End Switch Exit Routine (DFSFEBJ0)

The Front-End Switch (FES) exit routine allows you to keep the input terminal in response mode while it is waiting for the reply from the processing system for messages entered in an IMS system by a front-end switchable VTAM node and processed in another system (such as IMS or CICS).

In this Chapter:

About This Routine

During system definition, you specify the FES exit routine on the COMM macro with the FESEXIT parameter, and you specify which VTAM nodes can do front-end switching.

The connection between intermediate IMS systems must be through Intersystem Communication (ISC), although connections with non-IMS back-end systems can be any VTAM protocol that IMS supports, such as SLUTYPEP or SLUTYPE2. IMS-to-IMS and IMS-to-non-IMS interconnections are referred to as intermediate/back-end or IBE links, and front-end systems are referred to as FE systems.

Front-End Switch is not related to Multiple Systems Coupling (MSC), and cannot be used with MSC for the processing of the same transaction. Front-End Switch is designed to connect an IMS network to non-IMS systems, whereas MSC is used for homogeneous IMS networks.

Attributes of the Routine

You must code the exit routine for AMODE=31. You can define the RMODE as ANY.

Terminal Input Processing

The Front-End Switch exit routine gains control from an IMS system when the first segment of an input message is received before IMS determines the destination of the message. All input from FES-capable nodes and from ISC links are processed by this exit routine. By the time the message arrives, it already has been edited by routines such as Basic Edit, ISC (Intersystem Communication), and MFS Edit. Both MFS Edit and Basic Edit can remove characters that have a value less than X'41'.

For a diagram of the relationships among the front-end system, the intermediate system, and the back-end system with regard to message switching, see Figure 16 on page 270.

This exit routine can do any of the following:

- Indicate a destination change for an input message to an IBE destination or local transaction program defined in this IMS system. Changing the destination forces the originating terminal to be in response mode. (Front-end system processing.)

- Indicate a destination change for an input message from an IBE link to another IBE destination or to a local transaction program defined in this IMS system. (Intermediate system processing.)
- Define a transaction code that can be initiated when a specified time interval expires after switching the message. (Timeout processing.)
- Specify the message that can be sent directly to the input terminal for timeout processing.
- Provide IMS with additional routing information to expand the original message for any IBE system.
- Specify the name of a transaction program (full-function response mode or fast path) that processes or logs input messages due to user exit routine failures detected in other than the original terminal input (for example, ISC input).

The exit routine must provide additional routing information to identify the reply to this message when it comes back to the IMS front-end system. The user can tell IMS to remove the added information before the reply message is sent to the original terminal.

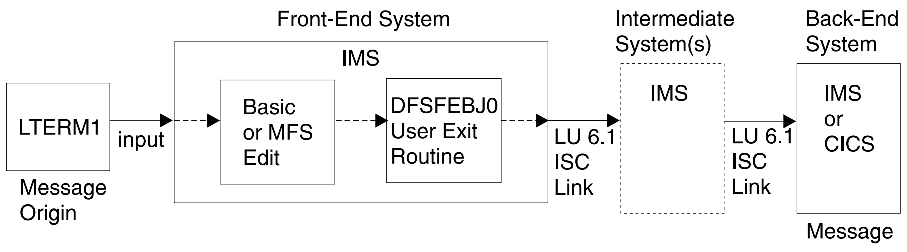


Figure 16. Message Flow with the Front-End Switch Exit Routine

In Figure 16, the reply path is not shown to keep the diagram simple. The reply would usually follow the same path back through the intermediate system(s) to the front-end, and then to the originating terminal.

IBE Input Processing

The exit routine takes control of each message that comes via an ISC or FES-defined link. It is the user’s responsibility to correlate the reply message to a previously switched input message. A reply to an input message, when received from another system, is treated by IMS as an input message.

The exit routine at this point can:

- Analyze the message text.
- Copy the LTERM name from the message text into the FEIBLTRM field.
- Copy the message identifier from the message text into the FEIBUNID field.
- Specify a destination for a late reply message in the FEIBLDST field.
- Tell IMS to remove the routing data from the message by specifying a length > 0 in the FEIBULNG field.
- Set the FEIBRPQ1 indicator if the reply message has to be sent directly to the original input terminal.
- Indicate the change of the destination code to a local transaction code (full-function non-response mode) in the FEIBNDST field.
- Set FEIBRPN to an error processing program (ERP) name that receives the input message if errors are detected in the verification of the exit parameters. An error message appears on the MTO of the system detecting the error.

Restrictions

The following restrictions apply to the Front-End Switch exit routine:

- The FES function can be used with the COMM macro statement specifying OPTIONS=BLKREQD or NOBLANK. However, you must specify a blank following the transaction code regardless of the option specified.
- If the back-end or intermediate system detects an error for an input transaction, the error message may not be sent back to the input terminal. It is sent to the MTO of the system detecting the error. It also can be sent back over the IBE session that sent the original input, or the input message can be sent to an ERP, if one is specified.

  If an error is sent over the ISC session, IMS will CLSDST the session thus making the error more visible and keep future ones from occurring. This may be valuable during a debugging period of a new FES exit or application; however, it may prove bothersome during production time. To avoid this, specify a FEIBERPN when processing input from an ISC session and develop an application to log or process these errors should they occur.
- Conversational transactions are not supported.
- If the front-end system is part of an XRF complex, the terminal operator may not get the reply to a switched message in case of a takeover even if the reply comes in time. The terminal receives an IMS message instead.
- For a local transaction defined as full-function, nonresponse mode, the exit routine switches a transaction (TXNA) to a local transaction (TXNB) and turns on the timer facility. TXNB executes locally and replies to the originating terminal. However, the terminal is left in response mode. When the timeout transaction processes, a response is sent to the terminal, which resets the response mode.
- If the back-end system is non-IMS, the reply message that the back-end system sends to IMS must be asynchronous (nonresponse) and expect no counter-response from IMS. You can do this in one of two ways:
  - End the response with an end bracket (EB).
  - Append the FMH6 SCHEDULER header to the FMH5 header at attach time, and use a change direction (CD) indicator.

  Related Reading: For more information, see *IMS/ESA Administration Guide: Transaction Manager*.

Front-End Interface Block

A Front-End Interface Block (FEIB) is created for each FES capable terminal. The FEIB is used to communicate between the Front-End Switch exit routine and IMS.

For a VTAM node (excluding ISC) defined as FES capable (by an OPTIONS=FES on the TERMINAL, or TYPE macro, or ETO logon descriptor), the FEIB is allocated when the session has been established. The block is released when the VTAM session terminates and no reply for an FES message is outstanding.

Related Reading: For more information on the Extended Terminal Option (ETO) feature, see *IMS/ESA Administration Guide: Transaction Manager*.

The interface block is also allocated for each ISC parallel session. This is done automatically without special system definition at LOGON or OPEN DEST time. The interface block is destroyed at LOGOFF time, at CLOSE DEST time, or at session failure.

If the exit routine is not defined in the system or if the VTAM node is not defined as FES capable, the FEIB will not be allocated.

Register 1 upon entry to the exit contains the address of the interface block.

The FEIB layout is in Figure 17. The FEIB usage is shown in Table 56 on page 274.

```
*-----------------------------------------------------------------------*
*      FEIB - FRONT END MESSAGE SWITCH INTERFACE BLOCK DSECT            *
*-----------------------------------------------------------------------*

FEIB     DSECT
FEIBIFLG DS    X                     USER EXIT INPUT FLAGS
FEIBISC  EQU   X'80'                 MESSAGE FROM AN ISC LINK
*        EQU   X'40'                 RESERVED BY IBM
*        EQU   X'20'                 RESERVED BY IBM
*        EQU   X'10'                 RESERVED BY IBM
*        EQU   X'08'                 RESERVED BY IBM
*        EQU   X'04'                 RESERVED BY IBM
*        EQU   X'02'                 RESERVED BY IBM
*        EQU   X'01'                 RESERVED BY IBM
FEIBOFLG DS    X                     USER EXIT OUTPUT FLAGS
FEIBRPQ1 EQU   X'80'                 QUEUE RESPONSE TO ORIG DEVICE
*                                    ELSE QUEUE SMB NAMED IN FEIBNDST
FEIBERP  EQU   X'40'                 ON TIMEOUT CALL ERP, ELSE ERR MSG
*        EQU   X'20'                 RESERVED BY IBM
FEIBTMED EQU   X'10'                 TIME RESPONSE WITH SYSDEF VALUE
*        EQU   X'08'                 RESERVED BY IBM
*        EQU   X'04'                 RESERVED BY IBM
*        EQU   X'02'                 RESERVED BY IBM
*        EQU   X'01'                 RESERVED BY IBM
FEIBMSGN DS    H                     TIMEOUT ERROR MESSAGE NUMBER
*                                    ONLY USED IF FEIBERP OFF
FEIBLTRM DS    CL8                   LTERM NAME OF ORIGINAL TERMINAL
*                                    ONLY AVAILABLE IF FEIBISC OFF
FEIBMSG  DS    A                     POINTER TO INPUT MESSAGE BUFFER
FEIBUNID DS    F                     UNIQUE ID NUMBER (FULL WORD BIN)
FEIBNDST DS    CL8                   NAME OF NEW DEST TO QUEUE MESSAGE
FEIBERPN DS    CL8                   NAME OF ERP TO CALL ON TIMEOUT
*                                    ONLY USED IF FEIBERP ON
FEIBLDST DS    CL8                   NAME OF DEST TO QUEUE LATE MESSAGE
FEIBULNG DS    H                     LENGTH OF DATA IN USER AREA
FEIBUSER DS    CL40                  USER AREA FOR DATA TO PREFIX MSG
*                                    ONLY USED IF FEIBULNG > 0.
FEIBIMID DS    CL4                   IMS IDENTIFIER
FEIBTIME DS    H                     TIMEOUT INTERVAL (SECONDS)
FEIBPRN DS     CL8                   PRIMARY RESOURCE NAME ADDED
                                     TO USER DATA BY ISC EDIT
```

Figure 17. FEIB DSECT from ICLI FEIUBASE=0

Description of the FEIB Fields

FEIBIFLG	**INPUT FLAG**	
	FEIBISC (bit 0)	on: message is from an ISC link
		off: message is from an FES capable device
	bits 1-7	reserved
FEIBOFLG	**OUTPUT FLAGS**	

	FEIBRPQ1 (bit 0)	on: reply message has to be sent directly to the original input terminal
		off: reply message has to be sent to an SMB named in FEIBNDST
	FEIBERP (bit 1)	on: on timeout, schedule the SMB named in FEIBERPN
		off: on timeout send text of error message defined in FEIBMSGN to the original input terminal (only used if FEIBTMED is ON.)
	FEIBTMED (bit 3)	on: release terminal from response mode when the timeout value is exceeded
		off: timeout facility is not used for this message
	FEIBDELT (bit 4)	on: defer timeout facility until FP sync–point
		off: timeout facility will be activated immediately at input message processing (only used if FEIBTMED is ON)
	bits 2,5-7	reserved
FEIBMSGN	User message number from table (DFSCMTU0) which is sent to the original input terminal in the case of a timeout. The message number can only be specified if the FEIBERP bit is off. Values range from 1-999. (Binary Number.)	
FEIBLTRM	Logical Terminal Name (LTERM) of the input terminal. For a reply message, DFSFEBJ0 must store the LTERM name into this field, padding with blanks on the right. (For more information, see “Routing Information” on page 275.)	
FEIBMSG	Pointer to the DC buffer containing the input message.	
FEIBUNID	Unique message identifier is only available if the FEIBISC bit is off on input to the exit routine. The exit routine must store the unique identifier (a binary number) into this field. (For more information, see “Routing Information” on page 275) for a reply message.	
FEIBNDST	New destination name for the message. This identify an IBE destination or a transaction code. (Blank padded on right.)	
FEIBERPN	Error processing program name (transaction code) to be scheduled in the case of a timeout. The FEIBERP bit must be set on if the program name is specified. (Blank padded on right.) This field is also used to specify an optional ERP if the input is from an IBE session. In this case, FEIBERP need not be set, and the ERP is scheduled with the input from the IBE session.	
FEIBLDST	Transaction name that is scheduled when a reply message arrives after timeout. (Blank padded on right.)	
FEIBULNG	This field must contain the length of the user data for an input message. It is used by IMS to expand the original message. This field can contain the length of the user data to be removed by IMS from the reply message for an output message.	
FEIBUSER	User data area for routing information that IMS uses to expand the message. This field is used for input messages only if the FEIBULNG field is greater than zero.	
FEIBIMID	At input to the exit routine, this field contains the identifier for the IMS system as specified during system definition on the IMSCTRL macro.	
FEIBTIME	Timeout interval override (in seconds). This field is used to override the system Front-End-Switch timeout value as supplied on the COMM macro. If a value of 0 is in this field, the system default override value is used.	
FEIBPRN	At input to the exit routine, this field contains the primary resource name that was added to the user data by ISC edit.	

Table 56 on page 274 shows the FEIB usage.

Table 56. FEIB Usage

Entry Name and Data Type	Input Message Processing						Reply Message Processing					
	Front-End System		Back-End System		Intermediate System		Front-End System/ph>		Intermediate System		Back-End System	
	In	Out	In	Out	In	Out	In	Out	In	Out	In	Out
`FEIBMSG DS A`	X		X		X		N/A	N/A	X		X	
`FEIBLTRM DS CL8`	X		X		X		N/A	N/A	X		X	X
`FEIBERPN DS CL8`		X		X			N/A	N/A		X		X
`FEIBMSGN DS H`		X					N/A	N/A				
`FEIBNDST DS CL8`		X		X			N/A	N/A		X		X
`FEIBUNID DS F`	X						N/A	N/A				X
`FEIBLDST DS CL8`							N/A	N/A				X
`FEIBULNG DS H`		X					N/A	N/A				X
`FEIBUSER DS CL40`		X					N/A	N/A				
`FEIBISC EQU BIT`	X(0)		X(1)		X(1)		N/A	N/A	X(1)		X(1)	
`FEIBRPQ1 EQU BIT`							N/A	N/A				X
`FEIBERP EQU BIT`		X					N/A	N/A				
`FEIBTMED EQU BIT`		X					N/A	N/A				
`FEIBIMID DS CL4`	X		X		X		N/A	N/A	X		X	
`FEIBTIME DS H`		X					N/A	N/A				
`FEIBPRN DS CL8`	X		X		X		N/A	N/A				
`Return Code (R15)`	2-New Dest from FE		6-New Dest from IBE		0-Nothing		N/A		6-New Dest from IBE		8-Reply	
Note: X(0) = off	X(1) = on											

Input and Output

Table 57 and Table 58 on page 275 show the input fields, which are stored by IMS and the output fields, which are stored by the exit routine.

Table 57. FES Data Flow for Input Message Processing

System	Input	Output
Front-End Systems	FEIBLTRM (CL8) FEIBMSG (A) FEIBUNID (F) FEIBIMID (CL4) FEIBPRN (CL8) Flags: FEIBSC	FEIBERPN (CL8) FEIBMSGN (H) FEIBNDST (CL8) FEIBUSER (CL40) FEIBULNG (H) FEIBTIME (H) RC= 0-Nothing 2-New Dest from FE Flags: FEIBERP, FEIBTMED

Table 57. FES Data Flow for Input Message Processing (continued)

System	Input	Output
Intermediate System	FEIBMSG (A) FEIBIMID (CL4) FEIBLTRM (CL8) FEIBPRN (CL8) Flags: FEIBSC	FEIBNDST (CL8) FEIBERPN (CL8) RC= 6-New DEST from IBE 12-Table Error Flags: N/A
Back-End System	FEIBMSG (A) FEIBIMID (CL4) FEIBLTRM (CL8) FEIBPRN (CL8) Flags: FEIBISC	RC= 0-Nothing Flags: N/A

Table 58 shows the input fields for reply message processing.

Table 58. FES Data Flow for Reply Message Processing

System	Input	Output
Front-End System	FEIBMSG (A) FEIBIMID (CL4) FEIBLTRM (CL8) Flags: FEIBISC	FEIBLTRM (CL8) FEIBUNID (F) FEIBLDST (CL8) FEIBNDST (CL8) FEIBULNG (H) FEIBERPN (CL8) RC= 0-Nothing 8-Reply 12-Table Error FEIBRPQ1 Flags:
Intermediate System	FEIBMSG (A) FEIBIMID (CL4)	FEIBNDST (CL8) FEIBERPN (CL8) **RC=** 0–Nothing 6–New Dest from IBE 12–Table Error **Flags:** N/A
	FEIBLTRM (CL8) Flags: FEIBISC	RC= 0-Nothing 6-New Dest from IBE 12-Table Error Flags:N/A
Back-End System	N/A	N/A

Routing Information

If the value of the FEIBULNG field is greater than zero, IMS adds the user data on an input message from an FE device to the input message between the old destination and the message text. You are responsible for the format and the contents of the routing information. Both MFS edit and Basic Edit can remove characters that have a value less than X'41'. As part of the routing information, the following is required:

- A unique identifier assigned to the input message from the originating terminal. This identifier must be sent with the user data to identify the reply to this message when it comes back to IMS. For messages being processed by either MFS or Basic Edit, the identifier value must be translated into unpacked format.
- The LTERM name of the originating terminal. IMS does not have access to the control blocks of the originating terminal when the reply to a switched message arrives. Therefore the exit routine must add the LTERM name of the originating terminal to the user data. This LTERM name is to be rerouted with the reply from the back-end system and must not be removed or changed by any intermediate system.

When the exit routine gains control from IMS on input of the reply message, it obtains the LTERM name and the unique identifier from the message text and stores them into the corresponding fields of the FEIB. IMS then determines the original input terminal and checks if timeout has already occurred. The destination of the message is determined by the result of this check.

If the timer has not expired, one of the following occurs:

- The message is sent directly to the original input terminal.
- The message is queued to a local transaction, which can cause a reply message to be sent to the originating LTERM using the I/O PCB.

Be aware that the TPCBTSYM field of the I/O (TPPCB) may contain the ISC LTERM name when the application does an ISRT reply back to the originating LTERM. This choice is decided by the exit routine.

If the timer has expired, the message is no longer expected at the original terminal, because it is already released from response mode. The message is then sent to the destination defined by the exit routine for late reply messages.

Besides required routing information, the routine can store additional information, such as a unique system identification throughout all connected systems.

Application programs processing FES messages must understand that the input message contains routing information which must be rerouted to the front-end system. The routing information in all the involved systems must be in agreement. The routing information in the input message must be included in the output message.

Message Expansion

Because the DC buffer is not large enough to store the routing information, use the FEIBUSER field of the FEIB. The length of the user data must be stored in the FEIBULNG field of the FEIB. The maximum length of user data is 40 bytes. IMS combines the original message with the user data and stores both into the new buffer. The new destination (FEIBNDST) is also stored into the new buffer.

Figure 18 shows the original and new buffer formats.

Original Buffer Format

LL	ZZ	Old_Dest	blank	Msg_Text

Buffer Format with RC=6

LL	ZZ	New_Dest	blank	Old_Dest	blank	Msg_Text

Buffer Format with RC=2

LL	ZZ	New_Dest	blank	Old_Dest	blank	User_Data	Msg_Text

Figure 18. Old and New Buffer Formats

New_Dest
New destination from FEIBNDST field

User_Data
User data from FEIBUSER field

The old destination and the new destination are both followed by a blank. You must lay out the routing information. After IMS has expanded the message, the routing information should precede the original message text.

Timer Facility

A timer facility controls each input message that is routed to a back-end system. When the specified time interval expires without a reply to the input message, the input terminal is released from response mode. The timeout value is specified during system definition on the COMM macro and can be overwritten by the FESTIM parameter on the IMS procedure, or by specifying a non-zero value in the FEIBTIME field during front-end processing of an input message. To make use of the timer, set the FEIBTMED flag in the FEIB. In addition, you must specify the action which has to be taken at timeout. This can be done by specifying either the name of a program that is to be given control (FEIBERPN field) or a message that is to be issued (FEIBMSGN field). The message number must be included in the user message table DFSCMTU0. See DFSCMTU0 for more information. The program can send a message to the input terminal using the I/O PCB. This response releases the terminal from response mode. The message text is directly sent to the input terminal if you define a message number.

If the reply comes in time, the timer request for the input message is canceled. No timeout can occur if you don’t set the FEIBTMED indicator. If no reply is received, the terminal is not released from response mode.

When switching to a local Fast Path transaction, the timeout facility can be deferred until Fast Path sync-point by setting the FEIBDELT flag.

FEIBRPQ1 Indicator

This indicator can only be set when a reply message has a return code of 8 in register 15. To have the message sent directly to the original input terminal, the FEIBRPQ1 indicator must be set in the FEIB. If you don’t set it, you have to store a new destination into the FEIBNDST field of the FEIB. IMS checks the indicator and sends the message, depending on the values in the FEIB.

If you change the destination code of an input message to a local transaction which sends a message across a link, the timer supervisor includes the elapsed time for the local transaction.

If the destination of a reply message is changed to a local transaction, the original input terminal is released from the timer supervisor before the local transaction is scheduled. If the transaction is not available or if the application program does not send an output message to the original input terminal, the terminal is not released from response mode.

Using IMS Callable Services with this Routine

To use IMS Callable Services with this routine, you need to issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service.

Related Reading: For details on using the initialization call, see “Step 2. Initializing IMS Callable Services (DFSCSII0)” on page 11.

Use the ECB found in Register 9 for IMS Callable Services. This exit is automatically linked to DFSCSI00 by IMS. No additional linking is required to use IMS Callable Services.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Content
1	Address of the FEIB. The FEIB contains all the information necessary for the exit to function. The exit routine must store additional information in the FEIB which is required for successful processing.
9	Address of ECB.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of exit routine.

Contents of Registers on Exit

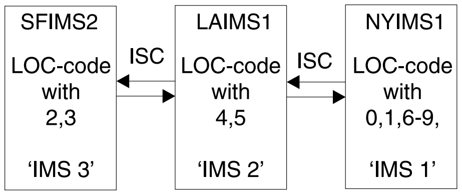
Before returning to IMS, the exit routine must restore all registers except for register 15, which must contain one of the following return codes:

Return Code	Meaning
0	No message switching
2	New destination from FE
6	New destination from IBE
8	Reply message
12	User table error

Example of the Front-End Switch Exit Routine (DFSFEBJ0)

In this example, three IMS systems are connected via ISC links. SFIMS2 acts as the front-end system, and LAIMS1 and NYIMS1 can act as a back-end system. In addition, LAIMS1 can act as an intermediate system.

Routing Scheme



In each system, the user can enter a transaction FESTX1. This is not defined as a transaction in the system, but is a special transaction code used by the sample exit routine that identifies this message as an FES transaction. The exit routine in the front-end system (SEIMS2) changes the transaction code to FESTX2, which must be defined in the system as a valid transaction.

There is an eight-digit location code (LOC-code) in the user data. The decision as to which system processes the transaction depends on this LOC-code. If the transaction is to be processed in another system, the exit routine changes the destination to LAIMS1 so that either LAIMS1 or NYIMS1 processes the transaction FESTX2.

The following location codes are defined:

System	LOCation-code
SFIMS2	20000000 - 39999999
LAIMS1	40000000 - 59999999
NYIMS1	00000000 - 19999999 60000000 - 99999999

The system that processes the transaction FESTX2 generates an output message containing the transaction code FESTX3 in front of the message text. As with FESTX1, this is not defined as a transaction in the system, but is a special transaction code used by the sample exit routine that identifies this message as a reply to an FES transaction. This output message has to be routed to the front-end system where the corresponding FESTX1 transaction was entered which is now the target system for the reply message.

In each system, two tables with routing information exist:

SFIMS2 -Table I			SFIMS2 -Table II[1]	
1st digit of LOC-code	Next system		Target system	Next system
0 1 4 5 6 7 8 9	LAIMS1 LAIMS1 LAIMS1 LAIMS1 LAIMS1 LAIMS1 LAIMS1 LAIMS1		LAIMS1 NYIMS1	LAIMS1 LAIMS1
			Note: [1] This table is used only if it is an intermediate system	

LAIMS1 -Table I			LAIMS1 -Table II	
1st digit of LOC-code	Next system		target system	Next system
0	NYIMS1		SFIMS2	SFIMS2
1	NYIMS1		NYIMS1	NYIMS1
2	SFIMS2			
3	SFIMS2			
6	NYIMS1			
7	NYIMS1			
8	NYIMS1			
9	NYIMS1			

NYIMS1 -Table I			NYIMS1 -Table II[1]	
1st digit of LOC-code	Next system		target system	Next system
2	LAIMS1		LAIMS1	LAIMS1
3	LAIMS1		SFIMS2	LAIMS1
4	LAIMS1			
5	LAIMS1			
			Note: [1] This table is used only if it is an intermediate system	

Description of Sample Exit Routine

The example in this section is based on the assumption that ISCEDIT is used for editing the messages going across ISC links. ISCEDIT removes the first data field of the message text on output to an ISC destination.

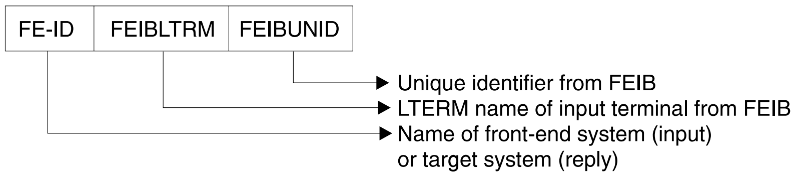
The exit routine is designed to run in each of the three systems without modifying the code. It has to process different tables with routing information for each system, and has to know the name of the owning system. This is obtained from the FEIBIMID field. In this example:

- NYIMS1='IMS1' back-end system
- LAIMS1='IMS2' back-end or intermediate system
- SFIMS2='IMS3' front-end system

The exit routine in each system must analyze the transaction code and the LOC-code in the message text:

- If the transaction code is FESTX1, and
  - Change the transaction code to FESTX2.
  - If the LOC-code is in table I:
    - Change the transaction code to FESTX2.
    - Change the destination to the corresponding destination from table I (FEIBNDST).
    - Set the FEIBTMED indicator on, if appropriate.
    - Set the FEIBERP indicator on, if appropriate.
    - Set the transaction code for ERP (FEIBERPN), if appropriate.

- Store the following routing information into the user area of the FEIB (FEIBUSER):


FE-ID
FEIBLTRM
FEIBUNID
Unique identifier from FEIB
LTERM name of input terminal from FEIB
Name of front-end system (input) or target system (reply)

  The FEIBUNID value, stored into the user area, is unpacked into zoned format to prevent MFS Edit or Basic Edit from removing characters less than X'41'.
- Set the user data length field to 24 (FIEBULNG).
- Set the RC=02 in register 15.

– Else Set RC=00 in register 15.

- If the transaction code is FESTX2 and the LOC–code is in table I:
  - Change the destination to the corresponding destination from table I (FEIBNDST).
  - Set the RC=06 in register 15.
- If the transaction code is FESTX3:
  - Analyze the routing information.
  - If the name of the target system in the routing information (FE–ID) is not the name of the owning system:
    - Change the destination to the corresponding destination from table II (FEIBNDST).
    - Set the RC=06 in register 15.
  - If the name of the target system is not table II, set RC=12 in register 15.
  - If the name of the target system in the routing information is the name of the owning:
    - Get the Lterm name from the routing information and store it into the interface block (FEIBLTRM).
    - Get a unique identifier from the routing information, change it from zoned to packed format, and store it in the interface block (FEIBUNID).
    - Set the transaction code for a message which comes too late (FEIBLDST).
    - Set the FEIBRPQ1–indicator.
    - Set the user data length field to 31 (FIEBULNG).
    - Set the RC=08 in register 15.
- In all other cases no action is taken by the exit routine.[2]

Using the Sample Front-End Switch Exit Routine (DFSFEBJ0)

For the latest version of DFSFEBJ), see the IMS.TMSOURCE library; member name is DFSFEBJ0.

2. In an IMS back–end system, which processes TX2, an application program generates the output message with TX3.

Chapter 34. Global Physical Terminal (Input) Edit Routine (DFSGPIX0)

This chapter describes the Global Physical Terminal Input edit routine. This routine is a user-written edit routine that is called before the IMS Basic Edit routine and performs the same functions as the Physical Terminal Input edit routine (DFSPIXT0).

About This Routine

If you write and include the routine in your system, IMS calls it for all terminals that do not have the Physical Terminal Input edit routine specified. By using the Global Physical Terminal Input edit routine instead of the Physical Terminal Input edit routine, you can eliminate the overhead associated with defining the edit routine for each terminal via system definition. For more information, see “Specifying the Routine” on page 284.

If the input message is processed by MFS, the Global Physical Terminal (Input) edit routine is not called. This edit routine is only called when a non-LU 6.2 message is entered from a terminal; it is not called when the message is inserted via a program-to-program switch.

Message segments are passed one at a time to the Global Physical Terminal (Input) edit routine, and the edit routine can handle them in one of the following ways:

- Accept the segment and release it for further editing by the IMS Basic Edit routine.
- Modify the segment (for example, change the transaction code or reformat the message text) and release it for further editing by the IMS Basic Edit routine. Your exit routine can make any required modifications because IMS has not yet performed destination or security checking.
- Cancel the segment.
- Cancel the message and request that IMS send a corresponding message to the terminal operator.
- Cancel the message and request that IMS send a specific message from the User Message Table to the terminal operator.

Table 59 shows the attributes of the Global Physical Terminal (Input) Edit exit routine.

Table 59. Global Physical Terminal Input Edit Routine Attributes

Attribute	**Description**
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSGPIX0.
Link editing	Follow the guidelines described in “Link-Editing the Routines” on page 5.
Including the routine	No special steps are required to include this routine.

Table 59. Global Physical Terminal Input Edit Routine Attributes (continued)

Attribute	Description
IMS callable services	To use IMS Callable Services with this routine, you must do the following: • Issue initialization call (DFSCSII0) to obtain the Callable Service token and a parameter list in which to build the function-specific parameter list for the desired callable service. For details on using the initialization call, see “Step 2. Initializing IMS Callable Services (DFSCSII0)” on page 11. • Use the ECB found in register 9 for the DFSCSII0 call. • Link DFSCSI00 with your user exit.
Sample routine location	IMS.TMSOURCE (member name DFSPIXT0). The sample is identical to the sample described in “Chapter 52. Physical Terminal (Input) Edit Routine (DFSPIXT0)” on page 369, because the two edit routines perform the same function. This routine performs the following functions: • Scans the input message segment for an expected format—TESTEXIT. • Generates return codes (XX) based on the input request (TESTEXIT,XX). • Verifies the user message number (YYY) if specified (TESTEXIT,XX,YYY). • Replaces TESTEXIT with ERROR if return code or message number is invalid and passes the segment to IMS (return code 0).

Bypassing Basic Edit

If the IMS application program supplies DFS.EDTN in the MOD name parameter for the output message, IMS bypasses the Basic Edit routine, except for transaction code and password validation.

Related Reading: For further information see “MFS Bypass for 3270 or SLU 2” in the “Application Programming Using MFS” chapter in *IMS/ESA Application Programming: Transaction Manager*.

The Physical Terminal Input edit routine must position the transaction code, and optionally the password, if the terminal is not operating in conversational or preset destination mode. The edit routine should detect errors and have IMS send a message to the terminal operator if the routine finds any errors.

IMS maintains a flag in the CTB (bit CTB6TRNI in the CTBFLAG6 field) to indicate when 3270 MFS bypass, nonconversational, no preset destination and first segment exist upon input to the Global Physical Terminal (Input) edit routine. This flag notifies the Global Physical Terminal Input edit routine that it can add a minimum of one and a maximum of 18 bytes to the front of the message segment for a transaction code and optional password. The minimum of one byte to be added to the front of the message segment consists of a one-byte transaction code. If NOBLANK is not specified at system generation, a minimum of two bytes is added to the front of the message segment, consisting of a one-byte transaction code and one blank, which is necessary as a separator. To add a transaction code and optional password, the exit routine can put a return code of 16 in register 15 and set register 1 to point to an LLZZ field followed by the data to be added.

Specifying the Routine

You must assemble and link-edit the edit routine into the IMS execution time library or user library concatenated in front of the IMS execution time library.

IMS calls the Global Physical Terminal Input edit routine (DFSGPIX0) for each terminal that does **not** have EDIT=(,YES) coded on the TERMINAL macro or ETO logon descriptor.

For terminals that do have EDIT=(,YES) specified on the TERMINAL macro or ETO logon descriptor, IMS calls the Physical Terminal Input edit routine (DFSPIXT0).

Related Reading:

- For more information on the TERMINAL macro, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.
- For more information on the Physical Terminal Input edit routine, see "Chapter 52. Physical Terminal (Input) Edit Routine (DFSPIXT0)" on page 369.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of Registers on Entry

Upon entry to the edit routine, all registers must be saved using the save area provided. The registers contain the following:

Register	Content
1	Address of the input message segment buffer. IMS editing has not been performed. The first two bytes of the buffer contain the segment length (binary length includes the 4-byte overhead). The third and fourth bytes of the buffer are binary zeros. The message text begins in the fifth byte of the buffer. If the device was defined with MFS support, but this message is not being processed by MFS, the first segment of the message has backspace error correction performed before entry to this edit routine. If escape (**) was entered by the terminal operator, the first two data bytes have been changed to binary zeros.
7	Address of CTB for the physical terminal from which the message was entered.
9	Address of CLB for the physical terminal from which the message was entered.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of edit routine.

The edit routine you supply can edit the message segment in the buffer pointed to by register 1.

You can reduce the length of the message segment to any size by replacing the length in the buffer with the appropriate value. The length field must appear in the same place at exit as at entry, and bytes 3 and 4 must not be changed.

Contents of Registers on Exit

Before returning to IMS, the edit routine must restore all registers except for register 1, which contains a message number if register 15 contains a value of 12; otherwise register 1 is ignored. Register 15 contains one of the following return codes:

Return Codes	Meaning
00	Segment is processed normally.
04	Segment is canceled.
08	Message is canceled and the terminal operator is notified.
12	Message is canceled, and the message identified by register 1 is sent to the terminal.
16	Insert the transaction code and optional password following the LLZZ pointed to by register 1. This return code is only valid for 3270 MFS bypass terminals. When the entering terminal is not a 3270 MFS bypass terminal, and the physical terminal input exit gives a return code of 16, IMS issues an error message, and the transaction code is not inserted in the message.

Any other return code causes the message to be canceled and the terminal operator to be notified.

Chapter 35. Greeting Messages Exit Routine (DFSGMSG0)

The Greeting Messages exit routine (DFSGMSG0) allows you to tailor how IMS handles messages issued during the logon and signon process. The exit also allows you to:

- Change the MFS Message Output Description (MOD) name without changing the message. (However, if the terminal is the Master Terminal and is master formatted, the request to change the MOD name is ignored.)
- Change the message without changing the MOD name.
- Send a null message (no data) for formatting purposes.

About This Routine

IMS builds a message based upon the calling module's request. This message, plus information useful to the exit and a buffer for returning an alternate message built by the exit, are passed as input to the exit. The exit indicates by a return code if the message built by IMS should be used, or if an alternate message has been returned and should be used. The message length returned must be at least five bytes (four bytes for the length field and a one-byte message).

Table 60 shows the attributes of the Greeting Messages exit routine.

Table 60. Greeting Messages Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSGMSG0.
Link editing	Follow the guidelines described in "Link-Editing the Routines" on page 5.
Including the routine	You can assemble the sample exit routine, or one that you write (using the standard IMS macro and copy files), and include it in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.RESLIB. If the Greeting Messages exit routine is included, IMS automatically loads it each time IMS is initialized.
IMS callable services	To use IMS Callable Services with this routine, you must do the following: • Issue an initialization call (DFSCSII0) to obtain the Callable Service token and a parameter list in which to build the function-specific parameter list for the desired callable service. For details on using the initialization call, see "Step 2. Initializing IMS Callable Services (DFSCSII0)" on page 11. • Use the ECB found at offset 0 of the Greeting Messages Exit parameter list. • Link DFSCSI00 with your user exit.
Sample routine location	IMS.TMSOURCE (member name DFSGMSG0). The sample exit uses the DFS3649 and DFS2467 messages built by IMS, but it converts the DFS3650 message to a single-segment message. You can also write your own exit routine.

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the exit routine.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	**Contents**
R1	Address of Standard Exit Parameter List (see Table 61)
R13	Save area address
R14	Return address to IMS
R15	Entry point address of exit routine

Table 61. Standard Exit Parameter List (Version 1)

Offset	Length	Description
+0	4	Address of standard exit parameter list version number
+4	4	Reserved
+8	4	Reserved
+12	4	Address of Greeting Messages Exit parameter list

Table 62 shows the greeting messages exit parameters.

Table 62. Greeting Messages Exit Parameter List

Offset	Length	Description
+0	4	ECB address.
+4	4	SCD address.
+8	4	Pointer to User Table.
+12	4	Address of parameter list for this exit. For additional details on the content and the format of these parameters, see the prolog in the sample routine.

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers except for registers 15, which contains the return code.

Register **Contents**

15 One of the following return codes:

Return Code	**Meaning**
00	Use the message built by IMS.
04	Use the message in the alternate buffer (single segment).
08	Use the message in the alternate buffer (multiple segment).
0C	Send a null message so that the device is formatted with the MFS format specified by IMS or returned by the exit.
10	Bypass password reverification. Valid only for message DFS3656A. **Related Reading:**See *IMS/ESA Messages and Codes* for further information.

Chapter 36. Initialization Exit Routine (DFSINTX0)

IMS calls the Initialization exit routine during initialization as a common Transaction Manager exit routine.

You can use the Initialization exit routine to create two user data areas that can be used by some of your installation's exit routines.

- General user data area

  The address of this user data area is passed as part of the IMS standard user exit interface. Any exit routine that uses this interface will have access to this data area (if it exists). The address of this data area is also passed as part of the non-standard interface to the following exit routines:

  Build Security Environment exit routine (DFSBSEX0)
  Command Authorization exit routine (DFSCCMD0)
  Greeting Messages exit routine (DFSGMSG0)
  Logoff exit routine (DFSLGFX0)
  Logon exit routine (DFSLGNX0)
  Non-discardable Messages exit routine (DFSNDMX0)
  Output Creation exit routine (DFSINSX0)
  Sign-Off exit routine (DFSSGFX0)
  Sign-On exit routine (DFSSGNX0)

  Other TM exit routines can address the user data table through SCDINTXP. Refer to the chapter for each exit routine for information on the routine's parameter list.

- LU 6.2 user data area

  The LU 6.2 user data area is not passed as part of the IMS standard user exit interface. It is passed as part the non-standard interface to the LU 6.2 Edit exit routine. Refer to "Chapter 43. LU 6.2 Edit Exit Routine (DFSLUEE0)" on page 321 for information on the routine' parameter list.

You can also use this exit routine to alter the setting for the Extended Terminal Option (ETO) feature. You can leave ETO activated or override the setting to indicate that ETO is not required, even if you previously requested it.

This exit is also used to enable password reverification. The IMS default processing is to disable password reverification. With password reverification, users signing on to VTAM terminals that change their password are prompted to verify the new password.

About This Routine

The Initialization exit routine is optional. If the exit is included in the system, IMS calls it before IMS loads the ETO descriptors and any exit routine that requires ETO to be active. If ETO is required for an exit routine, the documentation for the routine states that requirement. If the Initialization exit routine returns a return code indicating that ETO should not be made available, the ETO exit routines and descriptors will not be loaded. If this exit is not included in the system, IMS proceeds using the setting for the ETO= keyword that is specified as an EXEC parameter or in the DFSPBxx of IMS.PROCLIB.

The initialization exit routine can optionally enable password verification and an alternate ETO ALOT=0 option by setting the appropriate flags in the exit routine parameter list.

Attributes of the Routine

Table 63 shows the attributes of the Initialization exit routine.

Table 63. Initialization Exit Routine Attributes

Attribute	Description
IMS environments	This exit routine is used in all TM environments.
Naming convention	You must name this exit routine DFSINTX0.
Link editing	This exit routine must be reentrant.
Including the routine	If you want IMS to call the Initialization exit routine, include it in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.RESLIB. If the exit routine is included, IMS automatically loads it and calls it at initialization.
IMS callable services	DFSINTX0 can use callable storage services. To use IMS Callable Services with this routine, you must do the following: • Issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. For details on using the initialization call, see “Step 2. Initializing IMS Callable Services (DFSCSII0)” on page 11. • Use the ECB found at offset X'0' of the IMS Initialization exit parameter list. • Link DFSCSI00 with your user exit. For more information on using callable services refer to “Using IMS Callable Services” on page 8.
Sample routine location	IMS.TMSOURCE (member name DFSINTX0).

About User Data Areas

The user data areas can be used to provide access to user tables that can then be referenced by any user exit that has access to the data area. An example of the use of general user data area is for ETO. You can use the general user data area to define access limits for terminals or users by total number, department, time of day, or other criteria. You can also use the data area to define LTERM-to-user or user-to-terminal relationships to aid your installation logon and signon exit routine processes.

For APPC, you can use the LU6.2 user data area along with the LU6.2 User Edit exit routine to emulate MFS. To do so, the LU6.2 user data area is built by DFSINTX0 to hold a list of LTERM and MOD names available to the I/O PCB. IMS then passes the address of the LU6.2 user data area LU 6.2 Edit exit routine for input and output messages from a LU6.2 destination. The LU 6.2 Edit exit routine can use the list of LTERM names to redirect output to a non-LU6.2 destination, or the list of MOD names to format a message.

Related Reading: Refer to “Chapter 43. LU 6.2 Edit Exit Routine (DFSLUEE0)” on page 321 and *IMS/ESA Administration Guide: Transaction Manager* for more information on using MOD names and the LTERM interface.

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the exit routine.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
R1	Address of Standard Exit Parameter List (see Table 64)
R13	Save area address
R14	Return address to IMS
R15	Entry point address of exit routine

Table 64. Standard Exit Parameter List (Version 1)

Offset	Length	Description
+0	4	Address of standard exit parameter list version number
+4	4	Reserved
+8	4	Reserved
+12	4	Address of IMS Initialization exit parameter list

Table 65 shows the IMS initialization exit parameters.

Table 65. IMS Initialization Exit Parameter List

Offset	Length	Description
+0	4	CLB address
+4	4	SCD address
+8	4	0, as an indication that no user table exists
+12	4	0, as an indication that no LU 6.2 user table exists
+16	1	Input/Output Flag Byte **X'80'** **0** No password verification (default) **X'40'** **0** Default ETO ALOT=0 process

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers except for register 15, which contains the return code.

The address of the LU 6.2 user table created by this exit routine can be returned in the Initialization exit parameter list at +12. If zero, no LU 6.2 user table was created.

Register	Contents
15	One of the following return codes:

Return Code	Meaning
0	Initialization of IMS continues.
4	Regardless of ETO specification, ETO terminal support is not required. Message DFS3648 is sent to the system console. Setting RC=4 resets both ETO function and logon userdata support.
8	Regardless of ETO specification, ETO terminal support is not required but logon userdata is supported for static terminals. Message DFS3648 is sent to the system console. Setting RC=8 resets ETO function only.

Table 66. IMS Initialization Exit Parameter List

Offset	Length	Description		
+16	1	Input/Output Flag Byte		
		X'80'		
			0	No password verification (default)
			1	Enable password verification
		X'40'		
			0	Default ETO ALOT=0 process
			1	Alternate ETO ALOT=0 process

Chapter 37. Input Message Field Edit Routine (DFSME000)

This section describes how to write an Input Message Field edit routine. Because this routine is usually used with the Input Message Segment edit routine, you'll find references to both routines throughout the following paragraphs. For specific information about the Input Message Segment edit routine, see its description in"Chapter 39. Input Message Segment Edit Routine (DFSME127)" on page 305.

About This Routine

MFS application designers should consider the use of Input Message Field and Segment edit routines to perform common editing functions such as numeric validation or conversion of blanks to numeric zeros. Field and Segment edit routines can simplify programming by using standard field edits to perform functions that would otherwise have to be coded in each application program.

Table 67 shows the attributes of the Input Message Field Edit routine.

Table 67. Input Message Field Edit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSME000.
Link editing	Follow the guidelines described in "Link-Editing the Routines" on page 5. A Field edit routine must have a CSECT name of DFSMEnnn, where nnn is a number from 001 to 126 that corresponds with the routine number specified in the MFLD statement. The edit routine needs to be linked into the library specified by the USERLIB parameter of the IMSGEN Stage 1 macro before running the IMSGEN. The default for this parameter is IMS.RESLIB. The Field edit routine can only modify the data in the field created by MFS and must not cause any waits.
Including the routine	No special steps are required to include this routine.
IMS callable services	To use IMS Callable Services with this routine, you must issue an initialization call (DFSCSI00) to obtain the Callable Service token and a parameter list in which to build the function-specific parameter list for the desired callable service. **Related Reading:**For details on using the initialization call, see "Step 2. Initializing IMS Callable Services (DFSCSII0)" on page 11. Use the ECB found in register 9 for IMS Callable Services. This exit is automatically linked to DFSCSI00 by IMS. No additional linking is required to use IMS Callable Services.
Sample routine location	IMS.TMSOURCE (member name DFSME000).

Calling the Routine

Field edit routines are given control after MFS editing (before Segment edit routines, uppercase translation for all options, and null compression for option 1 or 2). The routine may validate or alter the data and pass a return code to MFS. MFS maintains the highest return code of all Field edit routines for each segment and passes that code to the Segment edit routine after all fields for that segment are edited.

Defining Edit Routines

Field edit routines are defined in the MID's MFLD statements in terms of a routine number and entry vector.

Routine numbers identify the routine to be used for this field/segment. Routine numbers range from 000 to 127. IMS-provided routines use numbers 000 (field edit, DFSME000) and 127 (segment edit, DFSME127).

If you are using both the Field edit and Segment edit routines with your IMS system, the Field edit routine should be assigned routine numbers that are lower than the numbers assigned for the Segment edit routine. Therefore, the Field edit number should be a decimal number greater than or equal to 0, and less than the default or specified value for the Segment edit routine number parameter. The default for the Field edit routine is 0.

An installation standard should be established regarding the assignment of routine numbers. For example, you could assign Field edit routines numbers in ascending sequence from 001 to 063 (and if you're using Segment edit routines as well, assign them numbers in descending sequence from 126 to 064).

These values are specified in the MFSEXIT= parameter in the COMM macro.

Related Reading: For details about this, see the description of the COMM Macro in the chapter on macros, in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Recommendation: It is recommended that the lower-numbered exit routines be Field exit routines, but the lower-numbered exit routines can be Segment exit routines, or a mixture of Field and Segment exit routines. It is recommended that the higher-numbered exit routines be Segment exit routines, but the higher-numbered exit routines can be Field exit routines, or a mixture of Field and Segment exit routines.

Entry vectors are passed to the edit routine when it is activated. Entry vector values can range from 0 to 255. The entry vector value can be thought of as an additional qualification of the routine to be activated. For example, routine number 025 may perform numeric validation of a field; entry vector 0 may replace leading blanks with zeros, and entry vector 1 may perform numeric validation.

If data is entered from the terminal in lowercase, the data is in lowercase when it is presented to the edit routine. If data in an input segment is in nongraphic form, GRAPHIC=NO should be specified in the SEG statement to prevent null compression and uppercase translation. A valid byte value of a binary field could be equivalent to a null character (X'3F') or some lowercase alphanumeric (for example, a=X'81'). In this case, GRAPHIC=NO should be specified.

Related Reading: For a description of which characters MFS considers graphic, see the SEG statement section in the *IMS/ESA Application Programming: Transaction Manager*.

Considering Performance

When Field and Segment edit routines are used, extra processing occurs in the IMS control region and, if used extensively, a measurable performance cost is incurred.

At the same time, these edit routines can improve performance by reducing processing time in the message processing region, by reducing logging and queuing time, and by allowing field verification and correction to be accomplished without scheduling an application program. Efficiency of these user-written routines should be a prime concern. Because these routines execute in the IMS control region, an abend in the edit routine causes the IMS control region to abend.

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the exit routine.

Contents of Registers on Entry

Upon entry, the edit routine must save all registers using the provided save area. The registers contain the following:

Register	**Contents**
1	Address of parameter list.
9	Address of CLB/ECB.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of edit routine.

Description of Parameter List Format

IMS.GENLIB contains a DSECT of the parameter list addressed by register 1 (use COPY MFSFLDE) as follows:

Byte	**Contents**	
0	**Bit**	**Contents**
	0,1	Message formatting option: • 00 = option 1 • 01 = option 2 • 11 = option 3
	2	Zero (Field edit routine)
	3	Reserved
	4	1 if the first 2 bytes in the field contains attribute information
	5	1 if the field contains extended field attribute information
	6	Reserved
	7	Reserved
1	Zeros	
2	The number of reserved extended field attribute bytes in the field. These bytes appear immediately after the 3270 attribute bytes, if any.	
3	The entry vector in binary (0 to 255).	
4-7	The execute length (length-1) of the field as defined in the MFLD statement. If ATTR=YES is specified, this field contains (length-3).	
8-11	The field address after MFS editing (before uppercase translation and null compression for option 1 and 2 fields). If ATTR=YES is specified, this is the address of the first data byte after the two attribute bytes. For option 3, this is the address of the 2-byte field length, which begins the completed option 3 field.	

Contents of Registers on Exit

Before returning to IMS, the edit routine must restore all registers except for register 15, which must contain one of the following return codes:

Register	Contents
15	Return code value from 0 to 255

Chapter 38. Input Message Routing Exit Routine (DFSNPRT0)

The Input Message Routing exit routine is provided to allow you to change the destination name (TRANSACTION or LTERM) or the destination system identification (SID), or associated MSNAME, of the input message immediately after it is received by IMS from the input device or program. By changing the destination name, you can reroute the message to a different destination name or IMS system than how it was originally (SYSGEN) or is currently (/ASSIGN or /MSASSIGN) defined.

The Input Message Routing exit routine can change the destination name of an input message to any local transaction or LTERM destination. In a multiple system coupling (MSC) environment, it can also change the destination name to any remote transaction or LTERM. For transaction-type messages, it can also override the MSC destination (SID) that is currently assigned to the transaction in the input system, and cause the message to reroute to a different IMS MSC system.

In a sysplex environment, or other environment where transaction macros are cloned as local transactions across IMS systems, this exit routine can control the routing of transaction messages to the various IMS systems. This rerouting is done by returning a MSNAME or SID to IMS, causing the message to reroute (over the MSC link) to another IMS system for processing.

About This Routine

The Input Message Routing exit routine provides similar functions as the existing MSC Terminal Routing exit routine, DFSCMTR0.

Either exit routine is called, but not both. If both DFSNPRT0 and DFSCMTR0 exist, only DFSNPRT0 is called.

The exit routine is passed a parameter list with information fields, some of which can be changed to affect the routing of the message.

The routing techniques available are:

- Change the destination name in the area pointed to by PARMLIST+16. This causes the message to reroute to the new destination with no change in the original transaction code or data.
- Change the destination name in the segment pointed to by PARMLIST+12 (if the segment is passed to the exit routine). This does not cause rerouting, but changes the transaction code passed to the user application program.
- Change the destination name in both the area pointed to by PARMLIST+16 and in the input segment pointed to by PARMLIST+12. This routes the message to the new destination and changes the transaction code passed to the user application program.
- Return a MSNAME or SID in the area pointed to by PARMLIST+32 or +36. This causes the message to be rerouted to the IMS MSC system that the assigned MSNAME or SID points to.
- Return a MSNAME or SID, and change the destination name pointed to by the PARMLIST and/or in the message segment. This changes the destination name the message is routed to and routes the message to the selected IMS MSC system.

This exit routine is called after receiving messages from the following programs, devices, or clients:
- Any IMS supported terminal
- Any intersystem coupling (ISC) LU 6.1 device or program
- Any advanced program-to-program (APPC) LU 6.2 device or program
- Any open transaction manager access (OTMA) client

For LU 6.2 APPC input messages this exit routine is called when the APPC conversation is allocated and before the data is received.

For open transaction manager access (OTMA) input messages, this exit routine is called when the transaction message is received from the OTMA client.

For all other input messages from IMS supported programs and devices, this exit routine is called when the first segment of the message is received from the device or program.

In all cases, the destination of the message has not yet been determined and might be changed by this exit routine. For all messages destined to a transaction, the destination MSC system can be specified. The transaction still must be defined (either local or remote) in the input system.

This exit routine is not called for input multiple system coupling (MSC) link.

For transaction-destined messages, the exit routine might return an MSC MSNAME or system identification (SID). If the multiple systems coupling (MSC) feature is SYSGENED, this causes the message to reroute to the remote IMS system via the MSC link assigned to the MSNAME or SID. This permits transactions that are defined as local (or remote) to reroute to a different IMS system for processing, than where the transaction is currently assigned. The transaction must also be defined as local with the same characteristics in the rerouted-to IMS system. This reroute capability can be used for transaction load balancing across IMS MSC systems that are connected by MSC links.

For LU 6.2 input messages, this exit routine can change the destination to a local or remote transaction but not to a local or remote LTERM or to a CPI-C driven application program.

Table 68 shows the attributes of the Input Message Routing exit routine.

Table 68. Input Message Routing Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSNPRT0.
Link editing	Follow the guidelines described in “Link-Editing the Routines” on page 5. The Input Message Routing exit routine must be reentrant. The Input Message Routing exit routine must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.RESLIB. This exit routine is optional.

Table 68. Input Message Routing Exit Routine Attributes (continued)

Attribute	Description
Including the routine	IMS loads this routine as a standalone module during IMS initialization if the routine exists in IMS.RESLIB (or a library concatenated to IMS.RESLIB). It receives control Amode 31, Rmode Any.
IMS callable services	Refer to "Using IMS Callable Services with This Routine".
Sample routine location	IMS.TMSOURCE (member name DFSNPRT0).

Communicating with IMS

IMS uses the Input Message Routine Exit parameter list to communicate with this routine.

Using IMS Callable Services with This Routine

The Input Message Routing exit routine can access the control block fields and flags shown in Table 69 through IMS Callable Services.

Table 69. Control Blocks Accessible to the Input Message Routing Exit Routine

Control Block	Accessible Fields
CTB	CTBFLAG1 (CTB1CON, CTB1MAST, CTB1SUBP)
CTT	CTTDEVIC
SMB	SMBDEQCT, SMBENQCT, SMBNAME, SMBSTATS (SMBSRESP, SMBSMULT, SMBSNOQU, SMBNOSC, SMBLOCK, SMBSINQU, SMBSQERR), SMBFLAG1, SMBFLAG2 (SMB2DRRT, SMBFPPTX, SMBFPXCL, SMB2SMS, SMB2RMT), SMBFLAG3 (SMB3WFI), SMBPRIOR, SMBCLASS, SMBSPAL, SMBLMTCT, SMBCOUNT, SMBSIDER, SMBSIDL, SMBMXRGN, SMBTCMDA, SMBNOSHCK, SMBPDIRN, SMBRCTEN
PDIR	PDIRSYM, PDIRCODE, PDIROPTC, PDIRBAD, PDIRFLG3

To use IMS Callable Services with this routine, you must do the following:

- Issue an initialization call (DFSCSI00) to obtain the Callable Service token and a parameter list in which to build the function-specific parameter list for the desired callable service. For details on using the initialization call, see "Step 2. Initializing IMS Callable Services (DFSCSII0)" on page 11.
- Use the current ECB address found at offset 0 for the DFSCSII0 call.
- Link DFSCSI00 with your exit routine.

To use the IMS Callable Services for non-LU 6.2 messages, use the address of the ECB at +0 of the Input Message Routing parameter list. The exit routines are optional. See Table 70 for what exit routine is called for the different types of messages.

Table 70. DFSCMTR0 or DFSNPRT0

Message Type	DFSNPRT0 exists	DFSCMTR0 exists	Both Exit Routines exist
LU 6.2 message	Call DFSNPRT0	None	Call DFSNPRT0
OTMA message	Call DFSNPRT0	None	Call DFSNPRT0

Table 70. DFSCMTR0 or DFSNPRT0 (continued)

Message Type	DFSNPRT0 exists	DFSCMTR0 exists	Both Exit Routines exist
Other type message	Call DFSNPRT0	Call DFSCMTR0	Call DFSNPRT0

Contents of Registers at Entry

At entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
R1	Address of standard IMS user exit routine parameter list
R13	Save area address
R14	Return address
R15	Entry point address

Table 71 lists the standard IMS user exit parameters.

Table 71. Standard Exit Parameter List (Version 1)

Offset	Length	Description
+0	4	Address of standard exit parameter list version number
+4	4	Reserved
+8	4	Reserved
+12	4	Address of IMS Input Message Routing exit parameter list

Table 72 shows the IMS input message routing exit parameters.

Table 72. IMS Input Message Routing Exit Routine Parameter List

Offset	Length	Description
+0	4	A(ECB)
+4	4	A(SCD)
+8	4	FLAGS (see below)
+12	4	ZERO or A(MESSAGE SEG)
+16	4	A(DESTINATION NAME)
+20	4	A(DESTINATION LENGTH)
+24	4	VARIABLE (see below)
+28	4	VARIABLE (see below)
+32	4	A(MSC MSNAME FIELD)
+36	4	A(MSC SID FIELD)

The following are the contents of the Input Message Routing exit routine parameter list for normal input messages, LU 6.2 (APPC) input messages, and OTMA input messages:

Offset	Contents
+0	Current ECB address

+4	SCD address	
+8	Message flag	

00000000	Message originated from non-LU 6.2 and non-OTMA client
nn000000	Message originated from non-LU 6.2 and non-OTMS 'nn' value

nn	**Description**
00	Value on entry to DFSNPRT0
00	Value on exit from DFSNPRT0, process normal.
08	Value on exit from DFSNPRT0, process message only on this IMS. This option is only valid in a shared queues environment and for a local transaction.

00000004	Message originated from an LU 6.2 application program or device
00000008	Message originated from OTMA client

+12 Address of the first message segment if message originated from non-LU 6.2 or non-OTMA (PARMLIST+8 = 0). Zero if LU 6.2 or OTMA message. Message segment has the format:

```
LLZZDESTNAMEDATA
```

Where

LL	2-byte segment length. You must not change this length.
ZZ	Segment flags. Do not change.
DESTNAME	Destination name. DESTNAME length is parameter at PARMLIST+20.
DATA	Remaining data in the segment.

+16 Address of a 1- to 8-byte destination name field (DESTNAME), left justified and padded with blanks to 8 bytes. The exit routine can change this destination name. If this name is changed, the DESTNAME length pointed to by the PARMLIST+20 might need to be changed to reflect the length of the new destination name.

+20 Address of a 4-byte field with the length of the destination name. If the destination name pointed to by PARMLIST+16 is changed, this length should be changed to reflect the length of the new destination name.

+24 Address of a 1- to 8-byte input LUNAME, left justified and padded with blanks. If the input message is from a non-LU 6.2 program or device, or non-VTAM program or device, or non-OTMA client (such as a BTAM device) the address will point to 8 bytes of zeros.

For LU 6.2 messages, address of a 1- to 17-byte network qualified input LUNAME, left justified and padded with blanks to 17 bytes.

For OTMA messages, address of a field that contains two 8–byte names. The first is a TPIPE name, the second is a destination override name. Each name is left justified and padded with blanks to 8 bytes.

+28 Zero for non-LU 6.2 and non-OTMA messages

For LU 6.2 messages, address of LU 6.2 message information block mapped by DSECT DFSMSGRT

For OTMA messages, address of a 1- to 16-byte member name padded with blanks to 16 bytes

+32 Address of 8 bytes of zeros that can be changed to a MSC MSNAME if the DESTNAME field is a TRANCODE. The message will be rerouted to the MSC link where the MSNAME is assigned to and sent to the designated remote IMS system. No rerouting is done if the field is 0. The 8-byte MSNAME must be left justified and padded with blanks. The multiple system coupling (MSC) feature must be available to reroute the message.

+36	Address of 2 bytes of zeros that can be changed to a MSC system identification (SID). The message will be rerouted to the MSC link where the SID is assigned to and sent to the designated remote IMS system. The MSC feature must be available to reroute the message. the SID must be a valid value (defined as remote) between 1-255 and stored in the low order byte of the field. If both SID and MSNAME are returned, SID takes precedence. If an invalid MSNAME or SID is returned, the reroute request is ignored and the transaction is routed to the system currently assigned to process the transaction (either local or remote).

Contents of Registers at Exit

Before returning to IMS, the exit routine must restore all registers. Register 15 is always zero.

At exit, the parameter list can be:

Parameter	Contents
+12	Pointer to segment must be the same, however the DESTNAME in the segment can be changed
+16	Can point to the DESTNAME at entry, or a new DESTNAME changed by the exit routine. PARMLIST+20 points to the length of the new DESTNAME and must be changed to reflect the new length.
+20	Address of the DESTNAME length, which must equal the DESTNAME being returned by the PARMLIST+16.
+32	Either zeros or the 1- to 8-byte MSNAME to reroute the transaction message.
+36	Either zeros or the 2-byte SID to reroute the transaction message.

Rules and Restrictions

For IMS conversational transactions, the exit routine is called only for the first message that starts the IMS conversation. Subsequent continuation messages from the input terminal in conversation to the user application program bypass the exit routine. If the exit routine rerouted the message to a MSC MSNAME or SID when the exit was called for the first message, all conversation continue messages will be rerouted to the same MSNAME or SID. This includes messages rerouted to another transaction name via a deferred program-to-program switch initiated by the application program.

The exit routine is not called for program-to-program switches from or to non-conversational transactions, or to immediate program-to-program switches from conversational transactions.

Related Reading: See “Chapter 47. MSC Program Routing Exit Routine (DFSCMPR0)” on page 345 for information on routing messages from application programs.

When rerouting a transaction type message to a remote MSNAME or SID, the transaction must be defined in both the local IMS system where the exit routine is being called, and the remote destination system where the transaction message is being rerouted to. The following parameters on the TRANSACT macro must be coded identically on the local and remote systems:

```
MSGTYPE = MULTISEG/SINGLSEG
INQUIRY = YES/NO, RECOVER/NORECOV
SPA = (SIZE)
```

Related Reading: See *IMS/ESA Installation Volume 2: System Definition and Tailoring* for information on this macro and parameters.

Chapter 39. Input Message Segment Edit Routine (DFSME127)

This section describes how to write an Input Message Segment edit routine. Because this routine is usually used with the Input Message Field edit routine, you will find references to both routines throughout the following paragraphs.

Related Reading: For specific information about the Input Message Field edit routine, see its description in “Chapter 37. Input Message Field Edit Routine (DFSME000)” on page 293.

About This Routine

MFS application designers should consider the use of Input Message Field and Segment edit routines to perform common editing functions such as numeric validation or conversion of blanks to numeric zeros. Field and Segment edit routines can simplify programming by using standard field edits to perform functions that would otherwise have to be coded in each application program.

Table 73 shows the attributes of the Input Message Segment edit routine.

Table 73. Input Message Segment Edit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSME127.
Link editing	Follow the guidelines described in “Link-Editing the Routines” on page 5. A Segment edit routine must have a CSECT name of DFSMEnnn, where nnn is a number from 001 to 126 that corresponds with the routine number specified in the SEG statement. It must be stored in USERLIB before Stage 2 of IMS system definition is executed.
Including the routine	No special steps are required to include this routine.
IMS callable services	To use IMS Callable Services with this routine, you must issue an initialization call (DFSCSII0) to obtain the Callable Service token and a parameter list in which to build the function-specific parameter list for the desired callable service. **Related Reading:**For details on using the initialization call, see “Step 2. Initializing IMS Callable Services (DFSCSII0)” on page 11. Use the ECB found in register 9 for IMS Callable Services. This exit is automatically linked to DFSCSI00 by IMS. No additional linking is required to use IMS Callable Services.
Sample routine location	IMS.TMSOURCE (member name DFSME127)

Calling the Routine

Segment edit routines are given control when all of the MFS editing and editing by Field edit routines is complete for a message (before uppercase translation, but after null compression for messages using option 1 and 2, and after field sort for option 3 messages). Based on the return code received from Field or Segment edit routine, the Segment edit routine can:

- Continue processing.
- Modify the segment.
- Cancel the segment.

- Cancel the message and IMS will notify the operator using the message `DFS298 INPUT MESSAGE CANCELED BY MFS EXIT`.
- Return a predefined message to the terminal.
- Return the input message to the terminal.

Restriction: The following applies only to IMS releases with ETO. During the ETO dynamic terminal sign on process, the Input Message Segment edit routine cannot use return code 16 to return the input message to the terminal. This is because a valid output LTERM has not yet been established.

Considering Performance

When Field and Segment edit routines are used, extra processing occurs in the IMS control region and, if used extensively, a measurable performance cost is incurred. At the same time, these edit routines can improve performance by reducing processing time in the message processing region, by reducing logging and queuing time, and by allowing field verification and correction to be accomplished without scheduling an application program. Efficiency of these user-written routines should be a prime concern.

Defining Edit Routines

Segment edit routines are defined in the MID's SEG statements. Each routine is defined in terms of a routine number and an entry vector.

Routine numbers identify the routine to be used for this field or segment. Routine numbers range from 000 to 127. IMS-provided routines use numbers 000 (Field edit, DFSME000) and 127 (Segment edit, DFSME127).

If you are using both the Field edit and Segment edit routines with your IMS system, the Field edit routine should be assigned routine numbers lower than the numbers assigned for the Segment edit routine. Therefore, the Field exit number should be a decimal number greater than or equal to 0, and less than the default or specified value for the Segment exit routine number parameter. The default for the Field edit routine is 0.

An installation standard should be established regarding the assignment of routine numbers. For example, you could assign Segment edit routines numbers in descending sequence from 126 to 064 (and if you're using Field edit routines as well, assign them numbers in ascending sequence from 001 to 063).

These values are specified in the MFSEXIT= parameter in the COMM macro.

Related Reading: For details about this, see the description of the COMM Macro in the chapter on Macros in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Recommendation: It is recommended that the lower-numbered exit routines be Field exit routines, but the lower-numbered exit routines can be Segment exit routines, or a mixture of Field and Segment exit routines. It is recommended that the higher-numbered exit routines be Segment exit routines, but the higher-numbered exit routines can be Field exit routines, or a mixture of Field and Segment exit routines.

Entry vectors are passed to the edit routine when it is activated. Entry vector values can range from 0 to 255. The entry vector value can be thought of as an additional qualification of the routine to be activated. For example, routine number 025 may perform numeric validation of a field; entry vector 0 may replace leading blanks with zeros, and entry vector 1 may perform numeric validation.

If data is entered from the terminal in lowercase, the data is in lowercase when it is presented to the edit routine. If data in an input segment is in nongraphic form, GRAPHIC=NO should be specified in the SEG statement to prevent null compression and uppercase translation. A valid byte value of a binary field could be a null character (X'3F') or some lowercase alphanumeric (for example, a=X'81'). In this case, GRAPHIC=NO should be specified.

Related Reading: For a description of which characters MFS considers graphic, see the SEG statement section in *IMS/ESA Application Programming: Transaction Manager*.

Communicating with IMS

IMS uses the entry registers, parameter list, and exit registers to communicate with the edit routine.

Contents of Registers on Entry

Upon entry to the edit routine, all registers must be saved using the save area provided. The registers contain the following:

Register	Contents
0	Address of CLB.
1	Address of parameter list.
9	Address of CLB/ECB.
13	Address of save area. The edit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of edit routine.

Description of Parameter List Format

IMS.GENLIB contains a DSECT of the parameter list addressed by register 1 (use COPY MFSSEGE) as follows:

Byte	Contents
0	

Bit	Contents	
0, 1	Message formatting option: 00 = option 1 01 = option 2 11 = option 3	
2	1 (Segment edit routine)	
3		
	1	If this message can be routed back to the device by specifying return code 16. This bit is set on when the following conditions are met: • PAGDEL=YES or OPTIONS=(...,PAGDEL,...) is specified in the TERMINAL macro for this device. • The device has an output logical terminal. If the message contains a valid operator logical paging request, bit 3 can be set on. However, this message is not returned to the terminal if requested.
4-7	Reserved	

1,2	Zeros
3	The entry vector is binary (0to 255).
4-7	The maximum segment length.
8-11	The segment address.
12-15	The highest return code from the Field edits for this segment.
16-23	The next MOD name.

The Segment Edit routine can modify only the segment contents, the save area, and the next MOD name field of the parameter list. The MOD name field name should be changed when the edit routine returns the input message to the device. If the segment is option 1 or 2, the routine can set the segment length field to any value from 0 to the maximum segment length. The Segment Edit routine must not cause any waits.

Contents of Registers on Exit

Upon return to IMS, all registers must be restored except for register 15, which must contain one of the following return codes:

Return Code	Meaning
0	Continue processing.
4	Cancel this segment.
8	Cancel this message (IMS sends the message DFS298 `INPUT MESSAGE CANCELED BY MFS EXIT`).
12	Cancel this message and return to the user the message whose number is in register 1.
16	Return this message to the input device. This code is allowed only when bit 3 of byte 0 in the parameter list is set on.

All segments of a multi-segment message are edited before the message is returned to the device (return code 16); if return code 8 or 12 is specified for a segment other than the final one, the message is canceled immediately and the remaining segments are not edited.

In IMS releases with ETO, the Input Message Segment edit routine cannot use return code 16 during the ETO sign on process. This is due to the lack of a valid output LTERM.

The functions of this IMS-supplied routine are as follows:

Vector	Resulting Action
0	Converts blanks to zoned decimal zeros (X'F0').
1	Converts blanks to zoned decimal zeros (X'F0') and replaces non-zoned decimal characters with a question mark (?). If ? is inserted, the routine sets a return code of 8 and, if an attribute (ATTR) area is present, sets the CURSOR,HI attributes.
2	Converts the binary cursor address field to zoned decimal if its length is 4 bytes. If the field is not 4 bytes, a return code of 8 is set.
>2	Sets a return code equal to the entry vector (if the vector is greater than 2).

The functions of this routine are based upon the entry vector and the highest Field edit routine return code (FLD-RC) for the segment, and it only performs modifications of messages using formatting options 1 and 2. These functions are:

Input Vector	FLD-RC	Resulting Function Action	SEG-RC
0	<4	None.	0[1]
	>=4	Places EBCDIC return code in last 3 bytes of the segment.	0
1	<4	None.	0
	>=4	Places EBCDIC return code in last 3 bytes of the segment.	0
	<8	None.	4[2]
2	<4	None.	0
	=4<8	Places EBCDIC return code in last 3 bytes of the segment.	0
	>=8	None.	8[3]
3	<4	None.	0
	=4<8	Places EBCDIC return code in last 3 bytes of the segment.	
	>=8	None	16[4]
4	ANY	Sets FLD-RC as user message number.	12[5]

Notes:

1. To continue processing
2. To cancel this segment
3. To cancel this message
4. To send this message back to the entering terminal
5. To cancel this message and send the user message, whose number is in register 1, back to the entering terminal

Chapter 40. Logoff Exit Routine (DFSLGFX0)

This chapter describes how you can use the Logoff exit routine to perform processing that complements the Logon exit routine (DFSLGNX0) described in “Chapter 41. Logon Exit Routine (DFSLGNX0)” on page 315.

About This Routine

The Logoff exit routine must handle all non-MSC, non-LU 6.2 VTAM nodes with which IMS communicates. IMS calls the Logoff exit routine for all master terminal operator (MTO) logoffs, even if it did not call the Logon exit routine for the MTO at logon. (Keep this in mind if your installation maintains a logon count.) All attempts to log off of ACF/VTAM terminals cause IMS to call this exit routine.

Related Reading: For more information on the Extended Terminal Option (ETO) feature, see *IMS/ESA Administration Guide: Transaction Manager*.

Recommendation: Although the Logon exit routine and the Logoff exit routine are optional, if you include one, you should also include the other to perform any necessary cleanup operations.

Table 74 shows the attributes of the Logoff exit routine.

Table 74. Logoff Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSLGFX0.
Link editing	Follow the guidelines described in “Link-Editing the Routines” on page 5.
Including the routine	If you want IMS to call this exit routine, include it in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.RESLIB. If the Logoff exit routine is included, IMS automatically loads it each time IMS is initialized.
IMS callable services	To use callable services with this routine, you must do the following: • Issue an initialization call (DFSCSI00) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. For details on using the initialization call, see “Step 2. Initializing IMS Callable Services (DFSCSII0)” on page 11. • Use the ECB found at offset 0 of the Logoff user exit parameter list. • Link DFSCSI00 with your user exit.
Sample routine location	IMS.TMSOURCE (member name DFSLGFX0).

Extended Recovery Facility (XRF) Considerations

Each time IMS calls the Logoff exit routine, the exit routine receives information on the XRF status of IMS. IMS calls the exit routine if XRF tracking fails.

Resetting the Significant Status

You can use this exit to reset the significant status for a terminal in one of the following states:

Conversational

Exclusive
Test
Preset
MFS test
Response

A parameter passed to the exit routine indicates the status of the terminal or ETO user at sign off. All usersexcept ETO terminalscan reset the status in the output parameters.

Resetting Status When VTAM Manages Generic Resource Affinities

When you specify GRAFFIN=VTAM on the DFSDCxxx PROCLIB member, IMS automatically insures that all non-ISC terminal status conditions are reset for sessions using the IMS generic resource name before the Signoff or Logoff exit routine or before the next logon or signon to a failed and restarted IMS.

For **conversation mode**, IMS performs the equivalent of an `/EXIT` command for the conversation.

: Refer to the *IMS/ESA Operator's Reference* for information about the `/EXIT`command.

For **full-function response mode**, IMS sends the response mode reply asynchronously.

For **Fast Path response mode**, IMS resets the response mode and discards the output reply message when a transaction is:

- Scheduled in global mode
- Scheduled in local mode, and the output is available when IMS attempts to reset status

Note that, regardless of whether the Fast Path output reply message is available, the log record indicates that the dequeue (reset of Fast Path response mode) is a result of the GRAFFIN=VTAM option. The log record uses the new flag shown below:

```
FLDQFLG DS X                 FLAG BYTE
FLDQFFDQ0 EQU X'10'          DEQ VIA GRAFFIN OR USER EXIT
```

Session termination during local mode Fast Path input scheduling or processing does not prevent VTAM from releasing affinity. However, IMS cannot terminate the Fast Path process. Therefore, it does not reset response mode before the next session is initiated. Input attempted in this case causes a DFS2162 to occur. The behavior is the same as when you specify GRAFFIN=IMS.

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the exit routine.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
R1	Address of Standard Exit Parameter List (see Table 75)
R13	Save area address
R14	Return address to IMS
R15	Entry point address of exit routine

Table 75. Standard Exit Parameter List (Version 1)

Offset	Length	Description
+0	4	Address of standard exit parameter list version number
+4	4	Reserved
+8	4	Reserved
+12	4	Address of Logoff Exit parameter list

Table 76 lists the logoff exit parameters.

Table 76. Logoff Exit Parameter List

Offset	Length	Description
+0	4	Current ECB address
+4	4	SCD address
+8	4	Address of User Table
+12	4	Address of the STATUS_IN and STATUS_OUT vectors. The status vectors are mapped by the DFSSTCHK macro. For the contents of the STATUS_IN vector see “Contents of STATUS_IN”. For the contents of the STATUS_OUT vector see “Contents of STATUS_OUT”.

Contents of STATUS_IN

The input status vector is a two-byte field that indicates the significant status of a terminal when the exit routine is called. The second byte of the field is reserved. The first byte of the field contains a value that indicates the significant status as follows:

Value	Description
X'80'	Conversation
X'40'	Exclusive
X'20'	Test
X'10'	Preset
X'08'	MSF Test
X'04'	Response

Contents of STATUS_OUT

The output status vector is a two-byte field that indicates changes to the terminal’s significant status made by the exit routine. IMS uses the contents of STATUS_OUT as an indicator to exit a conversation and reset significant status. The default for this field is zeros, indicating that no significant status is reset.

The second byte of the field is reserved. The first byte of the field contains a value that indicates the significant status as follows:

Value	Description
X'80'	Exit conversation
X'40'	Reset exclusive
X'20'	Reset test
X'10'	Reset preset
X'08'	Reset MFS test
X'04'	Reset Response

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers except for registers 15

Register	Contents
15	Ignored by IMS in all cases

Chapter 41. Logon Exit Routine (DFSLGNX0)

This chapter describes the Logon exit routine. The Logon exit routine enables you to control the way logons are processed.

About This Routine

The exit routine must handle all non-MSC, non-LU 6.2 VTAM nodes (excluding MTOs at IMS initialization) with which IMS communicates. All attempts to log on to ACF/VTAM terminals if ETO is active cause IMS to call this exit routine.

Related Reading: For more information on the Extended Terminal Option (ETO) feature, see *IMS/ESA Administration Guide: Transaction Manager*.

Depending on your installation's needs, you can write the Logon exit routine to:

- Select the logon descriptor that you want IMS to reference when building the terminal control block structure for the logical unit (LU) that is logging on. (For more information, see "Selecting a Logon Descriptor" on page 316.)
- Create or modify the user data that you want IMS to pass to the Sign-On exit routine (DFSSGNX0). The user data can be entered as autologon data, with the `/OPNDST` command, or with the VTAM internal commands INITSELF or INITOTHER. Alternatively, the Logon exit routine can build the user data.
- Allow or disallow a logon attempt based on the maximum number of sessions, or manage logons according to the time of day, certain terminal names, or other criteria that you specify.
- Specify or override the autologoff (ALOT), autosignoff (ASOT), screen size, or model values.

The Logon exit routine is optional.

Recommendation: If you include this exit routine, you should also include the Logoff exit routine (DFSLGFX0) to perform any necessary cleanup operations.

Related Reading: The Logoff exit routine is described in "Chapter 40. Logoff Exit Routine (DFSLGFX0)" on page 311.

If you do not supply the Logon exit routine, logons proceed as usual with the logon descriptor chosen according to "Selecting a Logon Descriptor" on page 316.

Table 77 shows the attributes of the Logon exit routine.

Table 77. Logon Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSLGNX0.
Link editing	Follow the guidelines described in "Link-Editing the Routines" on page 5.
Including the routine	If you want IMS to call the Logon exit routine, include it in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.RESLIB. If the exit routine is included, IMS automatically loads it each time IMS is initialized if ETO=Y (after the Initialization exit routine, DFSINTX0, changed the ETO= keyword).

Table 77. Logon Exit Routine Attributes (continued)

Attribute	Description
IMS callable services	To use callable services with this routine, you must do the following: • Issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. For details on using the initialization call, see “Step 2. Initializing IMS Callable Services (DFSCSII0)” on page 11. • Use the current address ECB found at offset 0 for the DFSCSII0 call. • Link DFSCSI00 with your user exit.
Sample routine location	IMS.TMSOURCE (member name DFSLGNX0).

Extended Recovery Facility (XRF) Considerations

During XRF tracking mode, IMS calls the Logon exit routine in the alternate system when the terminal control blocks are created for an XRF type 1 session with an ETO terminal. If processing is on an XRF alternate system, IMS ignores the contents of register 15 on exit. The exit routine is called during XRF alternate tracking only for the logon of a class 1 terminal.

Selecting a Logon Descriptor

If the terminal control block structure already exists for the terminal that is logging on, no logon descriptor is needed, and IMS uses the existing terminal control block structure.

If no terminal control block structure exists for the terminal, you can write the Logon exit routine to select the logon descriptor, select a logon descriptor by using the LOGOND= keyword, or let IMS select the logon descriptor using the LU name or default descriptor. Figure 19 shows the search order IMS uses to select the logon descriptor. IMS selects the first valid logon descriptor that it finds and uses that logon descriptor to build the terminal control block structure. If IMS cannot find a valid logon descriptor, including the default logon descriptor, it rejects the logon request.

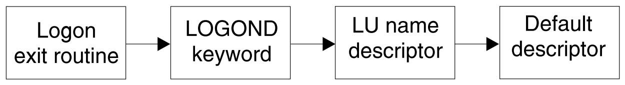


Figure 19. Logon Descriptor Search Order

If the exit routine supplies the name of a valid logon descriptor, IMS uses the logon descriptor associated with that name to build the terminal control block structure. If the Logon exit routine does not choose a logon descriptor, or if the exit routine is not included in the system, IMS uses the logon descriptor requested on the LOGOND= keyword (entering the keyword and descriptor as user data when you log on). If neither the exit routine nor the LOGOND= keyword identifies a valid logon descriptor, IMS searches for a logon descriptor with the same name as the logical unit (LU). If IMS cannot locate a logon descriptor with this name, IMS uses the default logon descriptor table shown in Table 78 to select the logon descriptor.

Table 78. Default Logon Descriptor Table

CINIT LUTYPE	CINIT TS	Default Logon Descriptor
X'06'	Not applicable	DFSLU61

Table 78. Default Logon Descriptor Table (continued)

CINIT LUTYPE	CINIT TS	Default Logon Descriptor
X'04'	Not applicable	DFSSLU4
X'02'	Not applicable	DFSSLU2
X'01'	Not applicable	DFSSLU1
X'00'	X'04'	DFSSLUP
X'00'	X'03'	DFS3270

IMS cannot generate DFSFIN or DFSNTO logon descriptors because of conflicting CINIT information. The wrong default logon descriptors are chosen for the FINANCE and NTO terminal types unless you do all of the following:

- Write the Logon exit routine so that it always supplies the appropriate logon descriptor name.
- Rename DFSFIN to DFSSLUP if no SLU P terminals exist.
- Rename DFSNTO to DFSSLU1 if no SLU1 terminals exist.

If you do not want dynamic logons for a certain LU type, delete the default logon descriptor for that type from the system, and be sure that the exit routine does not attempt to choose it.

Regardless of how the logon descriptor is selected, the descriptor must agree with the LUTYPE and TS fields (in the MODEENT macro of the VTAM mode table), or IMS rejects the logon request.

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the exit routine.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
R1	Address of Standard Exit Parameter List (see Table 79)
R13	Save area address
R14	Return address to IMS
R15	Entry point address of exit routine

Table 79. Standard Exit Parameter List (Version 1)

Offset	Length	Description
+0	4	Address of standard exit parameter list version number
+4	4	Reserved
+8	4	Reserved
+12	4	Address of user logon exit parameter list

Table 80 list the user logon parameters.

Table 80. User Logon Exit Parameter List

Offset	Length	Description
+0	4	Current ECB address.
+4	4	SCD address.
+8	4	Pointer to User Table.
+12	4	Pointer to the parameter list received from ACF/VTAM when application logon or SCIP bind exit routines are scheduled. If processing is on an XRF system, this value is zero.
+16	4	Pointer to 13-word parameter list.
+20	4	CLB pointer for the node trying to logon. If the node does not yet exist, this value is zero. The node always exists on an XRF system.

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers except for registers 15, which contains one of the following return codes:

Register	Contents
15	One of the following return codes:

Return Code	Meaning
0	LOGON accepted
4	LOGON rejected

Chapter 42. LU 6.2 Destination Exit Routine (DFSLULU0)

This exit routine is deleted from IMS Version 5. The process performed by this exit routine has been incorporated into DFSCMUX0, described in “Chapter 44. Message Control/Error Exit Routine (DFSCMUX0)” on page 327.

Chapter 43. LU 6.2 Edit Exit Routine (DFSLUEE0)

This section describes the LU 6.2 Edit exit routine. This routine enables you to edit input and output LU 6.2 messages for IMS-managed LU 6.2 conversations. It is also called if a message is inserted from an alternate PCB destined for an LU 6.2 destination. This exit routine is for use with standard IMS and modified IMS application programs. It is not called for CPI Communications driven application programs.

About This Routine

You can write the LU 6.2 Edit exit routine to:

- View the contents of a message segment and continue processing.
- Change the contents of a message segment and continue processing.
- Discard a message segment.
- Perform a DEALLOCATE_ABEND of the LU 6.2 conversation.

For input messages, IMS calls the LU 6.2 Edit exit routine for each message segment before the message segment is inserted to the IMS message queue. The exit routine can edit message segments as necessary before the application program processes the input message.

For output messages, IMS calls the LU 6.2 Edit exit routine for each message segment before the message segment is sent to the LU 6.2 program. The exit routine can intercept the data sent by the application program and edit it for the particular destination.

Related Reading: For more information on LU 6.2, see *IMS/ESA Administration Guide: Transaction Manager*.

Table 81 shows the attributes of the LU 6.2 Edit exit routine.

Table 81. LU 6.2 Edit Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSLUEE0.
Link editing	Follow the guidelines described in “Link-Editing the Routines” on page 5. The LU 6.2 Edit exit routine must be reentrant. The IMS-provided default exit routine specifies a return code of zero. If you write your own exit routine, replace the IMS default routine by link-editing the one you wrote into the IMS.RESLIB or including it in an authorized library in the JOBLIB, STEPLIB, or LINKLIB library concatenated in front of IMS.RESLIB.
Including the routine	No special steps are required to include this routine.
IMS callable services	This exit routine is not eligible to use IMS Callable Services.
Sample routine location	IMS.TMSOURCE (member name DFSLUEE0). This sample is a default exit routine, which IMS always calls for LU 6.2 messages processed under the DL/I call interface.

Changing a Message Segment

The LU 6.2 Edit exit routine can change the message length and contents, provided that it resets the message length field to reflect the new length. The exit routine can increase the message length by up to 256 bytes, but the total length (length field, flag field, and message) cannot exceed 32,767 bytes. If the message exceeds this limit, IMS truncates the message and issues DFS1967 to the master terminal operator (MTO) to indicate a message buffer overlay. The exit routine can reduce the message length without restriction.

Network-Qualified Names

Network-qualified LU names can be up to 17 bytes long.

Related Reading: For more information on network-qualified names, see *IMS/ESA Administration Guide: Transaction Manager*.

MOD Name Support for APPC

An LU 6.2 application program can send the LTERM and the MOD name in the first segment of the message. IMS saves the LTERM and MOD name in the I/O PCB.

At entry, IMS provides the address of the MOD name in the first segment of the message sent to the LU 6.2 Edit exit routine (DFSLUEE0). DFSLUEE0 checks the contents of the first message segment. If IMS finds the MOD name, it uses the MOD name to format the output message. If IMS finds the LTERM, it can use the LTERM to change the destination of the output.

Use the Initialization exit routine (DFSINTX0) to create the user table. This exit routine must pass the address of the user table to IMS, and IMS passes the address to DFSLUEE0.

Communicating with IMS

IMS uses the entry and exit registers and a parameter list to communicate with the exit routine.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of parameter list. The parameter list contains the following addresses. (For the format of the fields pointed to by the parameter list, see "Data Format of Parameters" on page 324.)

Bytes	Content
00-03	Address of a flag field indicating what type of message caused IMS to call the exit routine. This field contains one of the following flags (fixed length, right justified, padded with zeros):

0	Input message
4	Output message

Bytes	Content
04-07	Address of the area containing either the input or output message segment length, message flag, and message segment (variable length, left justified). The value in the length field includes the length field, flag field, and message.
08-11	Address of transaction code (fixed length, left justified, padded with blanks).
12-15	Address of LU name (fixed length, left justified, padded with blanks).
16-19	Address of user ID (fixed length, left justified, padded with blanks).
20-23	Address of return code, which is an exit parameter.
24-27	Address of LTERM (fixed length, left justified, padded with blanks).
28-31	Address of MOD name (fixed length, left justified, padded with blanks).
32-35	Address of user table, which is an entry parameter.
36-39	Address of message flag. (If bit zero of the message flag equals 1, it is the first segment.)

Register	Contents
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of exit routine.

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers. The registers contain the following:

Register	**Contents**
1	Address of parameter list (provided at entry). The parameter list contains the following addresses. (For the format of the fields pointed to by the parameter list, see “Data Format of Parameters”.)

Bytes	**Content**
00-03	Used on entry only. (For contents, see “Contents of Registers on Entry” on page 322.)
04-07	Address of the area containing the message segment length, message flag, and message segment (variable length, left justified). The value in the length field is the total length and includes the length field, flag field, and message. For information on the restrictions associated with changing a message’s length and contents, see “Changing a Message Segment” on page 322.
08-19	Used on entry only.
20-23	Address of the area for one of the following return codes from the exit routine. (IMS treats any other value as 0.)
24-27	Address of LTERM (exit parameter).
28-31	Address of MOD name (entry and exit parameter).
32-35	Address of User Table (entry parameter.
36-39	Address of message flag (Bit 0 = 1 then first segment) (entry parameter).

Return Code	**Meaning**
0	IMS performs the default action: continue processing.
4	Discard this message segment.
8	DEALLOCATE_ABEND the conversation.

Data Format of Parameters

Table 82 shows the data type, length, and format of the fields to which the parameter list (addressed by register 1) points.

Table 82. Format of Parameters

Bytes	Data Address	Parameter Use	Data Type	Data Length	Data Format[1]
00-03	Address of flag	Fixed length, right justified, padded with zeros	Input	4 bytes	*X'flag'*
04-07	Address of message segment length, message flag, and message segment	Variable length, left justified	Input and output	n bytes[2]	**LLZZ***message*
08-11	Address of transaction code	Fixed length, left justified, padded with blanks	Input	8 bytes	*code*bbbb
12-15	Address of LU name	Fixed length, left justified, padded with blanks	Input	17 bytes	*name*bbbb
16-19	Address of user ID	Fixed length, left justified, padded with blanks	Input	8 bytes	*user ID*bb

Table 82. Format of Parameters (continued)

Bytes	Data Address	Parameter Use	Data Type	Data Length	Data Format[1]
20-23	Address of return code	Fixed length, right justified, padded with zeros	Output	4 bytes	*X'code'*
24-27	Address of LTERM	Fixed length, right justified, padded with zeros	Output	8 bytes	*ltermname*
28-31	Address of MOD name	Fixed length, left-justified, padded with blanks	Input and output	8 bytes	*modname*
32-35	Address of user table	Variable length	Output	? bytes[3]	*usertablename*
36-39	Address of message flag	Fixed length	Output	1 byte	*X'code'*

Note:

[1]**ZZ** = flag field; **LL** = length field; bb = blanks; words in *italics* represent data values. The value in the length field **LL** includes the length field, flag field, and message.

[2]The exit routine can increase the message length by up to 256 bytes, but the total length cannot exceed 32,767 bytes.

[3]The length of this user table is determined by the user.

Chapter 44. Message Control/Error Exit Routine (DFSCMUX0)

This chapter describes the Message Control/Error exit routine. This routine lets you control transactions, responses, and message switches that are in error. The exit routine can request that IMS handle the messages that are in error, depending on the condition that led IMS to call the exit routine. The /DEQUEUE command supports the MSNAME keyword so that this control is extended to messages queued on Multiple Systems Coupling (MSC) links.

About This Routine

You can write the Message Control/Error exit routine to:

- Perform processing at MSC link start and link termination time that is unique to your installation, such as obtaining and freeing additional storage, and activating and deactivating a program.
- Reroute a message to a different local or remote transaction, local or remote LTERM, or an LU 6.2 destination. The target LTERM must be an existing LTERM; IMS does *not* dynamically create the LTERM, even if the Extended Terminal Option (ETO) feature is active. For more information, see "Rerouting Messages" on page 329. For more information on the ETO feature, see the *IMS/ESA Administration Guide: Transaction Manager*.
- Discard a message and send an informational message to the current master terminal operator (MTO) or input terminal indicating that the message is discarded.
- Suppress the /DEQUEUE command, or suppress the command and send an informational message to the entering terminal indicating that the command is suppressed.

A sample exit routine is available from the IMS library. The sample exit routine is the default routine. IMS calls the sample exit routine unless you replace it with your own version. The sample exit routine includes code that supports the following keywords on the /DEQUEUE command:

lterm

node

msname

luname plus *tpname*

The default action for this exit routine is to proceed with the /DEQUEUE command.

Table 83 shows the attributes of the Message Control/Error exit routine.

Table 83. Message Control/Error Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSCMUX0.
Link editing	Follow the guidelines described in "Link-Editing the Routines" on page 5. This exit routine must be reentrant. The sample exit routine is a default routine. If you write your own exit routine, you must link-edit it with the IMS control region RESLIB.
Including the routine	Include this routine by link editing it as described above.

Table 83. Message Control/Error Exit Routine Attributes (continued)

Attribute	Description
IMS callable services	This exit routine cannot use callable services.
Sample routine location	IMS.TMSOURCE (member name DFSCMUX0). The sample routine provided is completely compatible with the MSC error handling and /DEQUEUE command processing that exists for prior releases of IMS. You can ensure compatibility by including this sample exit routine logic in your customized version. The MSNB DSECT appears on IMS.TMSOURCE (member name MSNB).

Calling the Routine

IMS calls the Message Control/Error exit routine and sets an entry flag in the interface block as a result of one of the following:

- Link start.

  A RSTART LINK command is entered to start an MSC link or when the MSC link is started by the partner system (MSC environment only).
- Link termination.

  This exit routine is called at link termination time mainly when a PSTOP link command is entered from IMS, or the link is stopped by the partner IMS, for all access methods of MSC. Most errors (such as, invalid data, queue error, or access method) in MSC do not cause the link to be terminated.

  For MSC VTAM, the exit routine is also called in the following cases:
  - CLSDST/TERMSESS complete
  - Lost term error
  - Request canceled by CLSDST
  - Error during start up
  - Cleanup or Notify
  - Z-net or cancel
- Send error.
  - XCF send failed.
  - An invalid data block (send error) is detected during a transmission (MSC environment only). The sender must handle the message that is in error. You can write the exit routine to check if the link is down or stopped at this time. DFS2140 with reason code 2146 indicates a send error.
  - An LU 6.2 session failed while sending an output message to an LU 6.2 program. The exit routine can only reroute or discard the message. The default action is to discard the message.
  - A send to an LU 6.2 program is rejected with a deallocate or with a send error. The exit routine can only ask IMS to reroute or discard the message. The default action is to discard the message.

Restriction: When the exit routine discards a message from an LU 6.2 conversation because a send error occurred, the exit routine must not send an informational message to the originating LU 6.2 application. The informational message can be rejected for the same reason that the original message was rejected.

If a send error occurs while sending a reply from an IMS local conversational transaction or a Fast Path transaction to an LU 6.2 program, this exit routine is not called. If the reply is from a remote transaction or a local nonconversational transaction, this exit routine is called.

- Receive error

  An input message error (receive error) is detected by the receiver of a message (MSC environment only). The following messages indicate a receive error: DFS064, DFS065, DFS067, DFS076, DFS1959E, DFS2125, DFS2126, DFS2127, DFS2128, DFS2129, DFS2130, DFS2131, DFS2132, DFS2133, DFS2134, DFS2137, DFS2141, DFS2143, DFS2163, DFS2164, DFS2165, DFS2167, DFS2174, DFS2175, DFS2176, and DFS3470.
- A `/DEQUEUE` command with the specified *lterm*, *node*, *msname*, *luname* plus *tpname* and *tmember name* plus *tpipe name* keyword is entered. IMS calls the exit routine before processing each message on the queue.

Related Reading:
- See *IMS/ESA Operator's Reference* for information on commands and keywords.
- See *IMS/ESA Messages and Codes* for message descriptions.
- See “Message Control/Error Exit Interface Block (MSNB)” on page 332 and “Valid Flags and Default Actions” on page 336 for more information on IMS calls to this exit routine and possible message processing actions.

Rerouting Messages

The Message Control/Error exit routine enables you to reroute transactions, responses, and message switches that are in error. When you reroute a message to a different destination, that destination must be a local or remote transaction, a local or remote LTERM, an LU 6.2 destination, or OTMA *tmember* and *tpipename*. The new destination must be capable of processing the message.

Restriction: You cannot reroute a message to a CPI-C driven application program.

Definition: An LU 6.2 destination is the LU 6.2 application program and is always defined with the LU name, plus the TP name.

A message that is rerouted to a transaction (conversational or nonconversational) can include the interface block if your exit routine sets the MSX2QBK bit in the MSXFLG2 field of the MSNB interface block. If this bit is on, a new message (with the interface block included) is built and enqueued to the new destination. If this bit is off, the original message is enqueued. For more information on these fields, see “Message Control/Error Exit Interface Block (MSNB)” on page 332.

The format of the message depends on the message type and the new destination type as shown in Table 84. Each destination type is discussed in the sections following the figure.

Table 84. Rerouting Messages to New Destinations

Message Type	New Destination	Message Format (if MSX2QBK is turned on)
1. Conversational	Conversational transaction	SPA + interface block + message
2. Conversational	Nonconversational transaction	Interface block + unpacked SPA + message

Table 84. Rerouting Messages to New Destinations (continued)

Message Type	New Destination	Message Format (if MSX2QBK is turned on)
3. Nonconversational (but not message switch)	Nonconversational transaction	Interface block + message
4. Message switch	Nonconversational transaction	Interface block + message
5. Message switch	LTERM	Original message
6. All types	Luname, tpname	
7. OTMA	Transaction, lterm, luname + tpname, or OTMA member name + tpipe name	Interface block + message if the new destination is transaction or lterm, the message format rules for message types 1 through 5 are applicable.

Recommendation: If a message must be rerouted, it is recommended that you reroute it to a local nonconversational transaction to avoid further error. This nonconversational transaction is a special-purpose error processing transaction and can process all messages that are rerouted to it.

Attention: During the rerouting process, the original message is dequeued first, and then the newly built message is enqueued to the new destination. If a system failure occurs between the dequeue and enqueue processing, the message may be lost.

Rerouting to a Conversational Transaction

When a conversational message is rerouted to another conversational transaction, the scratch pad area (SPA) is the first segment, and the interface block is the next segment (if your exit routine sets the MSX2QBK bit). If you reroute a conversational transaction to a different conversational transaction, make sure that both transactions have the same SPA size.

Rerouting to a Nonconversational Transaction

When the new destination is a nonconversational transaction, the interface block is the first segment of the rerouted message (if your exit routine sets the MSX2QBK bit).

If the message is conversational, the segment following the interface block is the unpacked SPA and should be treated as a data segment by the new destination's application program. If the message is conversational or is in response mode (or both), it is the user's responsibility to end the conversation and take the input terminal out of response mode. One of the following can be done to end the conversation or take the terminal out of response mode:

- Enter the `/EXIT` command from the input terminal, if the keyboard is not locked.
- If the input terminal is a static terminal, from the MTO or system console of the input system, enter:
  –
  ```
  /DISPLAY CONVERSATION HELD NODE nodenameor
  /DISPLAY CONVERSATION BUSY NODE nodename
  ```
  (to determine the conversation ID)

  - `/STOP NODE nodename`
  - `/EXIT CONVERSATION conversation id NODE nodename`
  - `/START NODE nodename`

    (if appropriate)

  These commands can also be issued from an AOI program.

- If the input terminal was dynamically created using the Extended Terminal Option (ETO) feature, from the MTO or system console of the input system, enter:
  - `/DISPLAY CONVERSATION HELD USER username` or
    `/DISPLAY CONVERSATION BUSY USER username`

    (to determine the conversation ID)
  - `/STOP USER username`
  - `/EXIT CONVERSATION conversation id USER username`
  - `/START USER username`

    (if appropriate)

  These commands can be issued from an AOI program.

Related Reading: For more information on these commands, see *IMS/ESA Operator's Reference*.

Rerouting to an LTERM

When the new destination for a message is an LTERM and a message is rerouted from one physical terminal type to another, IMS rejects the message and issues an error message (such as DFS2078) if the new destination cannot handle the data.

Related Reading: For more information, see *IMS/ESA Messages and Codes*.

Communicating with IMS

IMS uses the entry and exit registers, and the MSNB interface control block to communicate with the exit routine.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the save area provided. The registers contain the following:

Register	Contents
1	Address of Message Control/Error exit interface block, MSNB. (For the format of the interface block, see Contents of Interface Block on Entry.)
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of exit routine.

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers. The contents of the interface block pointed to by register 1 can be different. (For the format of the interface block on exit, see “Contents of Interface Block on Exit” on page 333.)

Message Control/Error Exit Interface Block (MSNB)

The interface block for the Message Control/Error exit routine contains all of the information about the message. The entry flag (MSNFLG1) indicates the reason the exit routine is called, and the exit flag (MSXFLG1) determines what action will be performed when control is returned to IMS. MSNBSEG1 points to the first segment of the message. If the segment is a SPA, IMS unpacks it before passing control to the exit routine. The exit routine can place any information that it needs into the user work area (MSNBUSRA); IMS does not disturb the contents of this work area.

The Message Control/Error exit routine can **only** modify seven fields: MSNBRTPG, MSNBRTPN, MSNBDEST, MSNBRINF, MSNBUSRA, MSXFLG1, and MSXFLG2. All other fields are read-only. If the exit routine modifies MSNBDEST, it must modify MSNBRINF. If the exit routine modifies MSNBRTPG and MSNBRTPN, it must modify MSNBRINF. In addition, the exit routine can modify MSXFLG2 if the exit routine modifies MSNBDEST and MSNBRINF, or MSNBRTPG, MSNBRTPN and MSNBDEST. (For more information about these fields, see “Contents of Interface Block on Entry” on page 332.)

Contents of Interface Block on Entry

Table 85 shows the contents of key fields in the Message Control/Error exit interface block as they appear on entry.

Table 85. Key Fields of Interface Block on Entry

Byte	Field Name	Contents	
X'C'	MSNFLG1	**Entry Flag**	**Meaning**
		X'80'	MSC link start
		X'40'	MSC link termination
		X'20'	Send error detected
		X'10'	Receive error detected
		X'08'	/DEQUEUE command entered
X'D'	MSNFLG2	**Entry Flag**	**Meaning**
		X'80'	Message prefix error detected (invalid **ZZ** field)
		X'40'	Invalid data block detected

Table 85. Key Fields of Interface Block on Entry (continued)

Byte	Field Name	Contents	
X'E'	MSNFLG3	**Entry Flag**	**Meaning**
		X'80'	DEQUEUE NODE command entered
		X'40'	DEQUEUE LTERM command entered
		X'20'	DEQUEUE MSNAME command entered
		X'10'	DEQUEUE LUNAME TPNAME command entered
		X'08'	DEQUEUE TMEMBER TPIPE name entered
X'F'	MSNFLG4	**Entry Flag**	**Meaning**
		X'80'	Message is a transaction
		X'40'	Message is a message switch
		X'20'	Message is a response
		X'10'	SPA in the message
		X'08'	Response mode message
		X'04'	Conversation starting
		X'02'	Message switch from DFSAPPC
		X'01'	Message from APPC type message
X'1A'	MSNBOSID	Source SYSID (if MSC)	
X'1C'	MSNBDSID	Destination SYSID (if MSC)	
X'1E'	MSNBMGID	Error message number (if receive error)	
X'20'	MSNBORGN	Message origin source name [1]	
X'48'	MSNBDSNM	Final destination of message	
X'74'	MSNBRTPG	Length of TP name from /DEQ LU name TP name command	
X'76'	MSNBRTPN	TP name from /DEQ LU name TP name command	
X'B6'	MSNBDEST	• Node if /DEQ node command	
		• LTERM if /DEQ lterm command	
		• MSNAME if /DEQ msname command	
		• LU name TP name if /DEQ luname tpname command	
X'12A'	MSNBUSRA	User work area	

Note: [1] In an LU 6.2 conversation, when the outbound message is re-enqueued across restart, the message origin source name (MSNBORGN) is blank.

Contents of Interface Block on Exit

Table 86 on page 334 shows the contents of key fields in the Message Control/Error exit interface block as they appear on exit. The exit routine uses these fields to return information to IMS.

Table 86. Key Fields of Interface Block on Exit

Byte	Field Name	Contents	
X'70'	MSXFLG1	**Exit Flag**	**Meaning**
		X'00'	No message is involved. (Perform the default action, which is the same action as in the prior release.) You can modify the exit routine to perform: • Initialization processing (including external IMS System Services) at link start • Clean-up processing at link termination
		X'80'	Reroute the message to a different local or remote transaction, a local or remote LTERM, or an LU 6.2 destination. The exit routine must provide the new destination name in the MSNBDEST field, and set MSNBRINF to indicate an LTERM, a transaction, or an LU 6.2 destination.
		X'60'	Perform actions of both X'20' and X'40'.
		X'40'	Discard the message or proceed with the /DEQUEUE command.
		X'30'	Perform actions of both X'10' and X'20'.
		X'20'	If the exit routine selects this action, IMS sends an informational message: • If the /DEQUEUE command was entered, IMS sends DFS2185 to the entering terminal. • If IMS detected a receive error, IMS sends DFS2184 to the current MTO or input terminal. • If IMS detected a send error, IMS sends DFS2184 to the current MTO. If this action is selected by default and not by the exit routine, IMS sends an informational message: • On a send error, IMS sends DFS2140. • On a receive error, IMS sends the message number in the MSNBMGID field. This exit flag can be specified only in combination with exit flag X'10' or X'40'.
		X'10'	Suppress the /DEQUEUE command. The /DEQUEUE PURGE operation is terminated if the exit routine requests to suppress the command.

Table 86. Key Fields of Interface Block on Exit (continued)

Byte	Field Name	Contents
X'71'	MSXDFT1	**Exit Flag** / **Meaning** **X'00'** No message involved (link start or link termination) or the default action. **X'80'** Reroute message to a different destination. **X'40'** Discard the message or proceed with the /DEQUEUE command. **X'20'** Send error message to current MTO or input terminal. **X'10'** Suppress the /DEQUEUE command.
X'72'	MSXFLG2	**Exit Flag** / **Meaning** **X'80'** MSX2QBK field; include interface block in the message when rerouting to a different destination.
X'74'	MSNBRTPG	Length of rerouted TP name.
X'76'	MSNBRTPN	Rerouted TP name.
X'B6'	MSNBDEST	Destination name of local or remote transaction or local or remote LTERM, or reroute LU name or reroute netid.luname (left-justified, padded with blanks) if reroute the message.
X'E1'	MSNBRINF	**Exit Flag** / **Meaning** **X'80'** Destination is a transaction. **X'40'** Destination is an LTERM. **X'20'** Destination is a dynamic local LTERM. **X'10'** Destination of LU name plus TP name. **X'08'** Destination of OTMA member plus tpipe.
X'E9'	MSMFLG1	**Exit Flag** / **Meaning** **X'80'** Next segment is a SPA.
X'106'	MSNBMSG	Message area when error encountered in the interface module.
X'12A'	MSNBUSRA	User work area.

Logging the Interface Block

Two copies of the interface block are added to the existing X'6701' log record. The first copy is labeled “MSNB” and represents the interface block before IMS calls the Message Control/Error exit routine with the log record ID of CMEA. The second copy is labeled “USR MSNB” and represents the interface block after IMS calls the exit routine with the log record ID of CMEB. The X'6701' log record can be logged for informational reasons or to indicate an error in preparing to call the exit routine, or in performing the action(s) requested by the exit routine. The trace ID is CMEI. These log entries are forced entries for a send error, a receive error, and a

/DEQUEUE command, regardless of any trace options that are specified. For link start and link termination, the interface block is only logged if the trace option is in effect on the link or node involved.

Related Reading: For more information on this log record, see *IMS/ESA Diagnosis Guide and Reference*.

Valid Flags and Default Actions

IMS performs the default actions (specified in the MSXDFT1 field) if the exit routine returns control to IMS without modifying the exit flag field (MSXFLG1) in the interface block. IMS also performs the default actions if the exit routine requests an invalid exit flag, or if IMS encounters an error while trying to perform the action requested by the exit routine. If an invalid exit flag is requested, IMS sends error message DFS2184 to the current MTO, in addition to performing the default action.

Table 87 shows valid entry flags, exit flags, and default actions. The entry and exit flags are described in “Contents of Interface Block on Entry” on page 332 and “Contents of Interface Block on Exit” on page 333.

Table 87. Flags and Default Actions

Entry Flag (MSNFLG1)	Valid Exit Flags (MSXFLG1)	Default Action (MSXDFT1)
X'80'	X'00'	X'00'
X'40'	X'00'	X'00'
X'20'	X'00', X'40', X'60', X'80'	X'60' + stop MSNAME
X'10'	X'00', X'40', X'60', X'80'	X'60'
X'08'	X'00', X'10', X'30', X'40', X'80'	X'40'

Note: The default action for a send error (entry flag = X'20') includes STOP MSNAME. In addition, the default action for the DEQUEUE command is to proceed with the command. If you do not want these actions to take place, specify a different exit flag depending on the actions that you want to occur.

If any errors are encountered while IMS tries to perform the requested action, the action is ignored and the default action is performed. The MSNBMSG field of the interface block of the forced 6701 CMEI log record will contain one of the following brief descriptions that describe the error encountered, if applicable:

- No storage for message buffer
- Invalid destination for reroute
- Cannot reroute MSG switch to CONV
- Error while building rerouted MSG
- Reroute destination not found
- Cannot reroute CONV MSG to LTERM
- Cannot reroute non-CONV MSG to CONV

Chapter 45. Message Switching (Input) Edit Routine (DFSCNTE0)

This chapter describes the Message Switching (Input) Edit routine. Information about using a sample routine is provided at the end of this chapter.

About This Routine

A facility similar to the Transaction Code (Input) Edit is provided for message switching. The optional user-written routine, whose CSECT and load module name must be DFSCNTE0, is included in the system at IMS system definition time. Only one Message Switching edit routine can be specified for an IMS online control program. This routine is specified for inclusion with the online control program by specifying EDIT=(YES,...) in one or more NAME macro during system definition. This routine is called when a message is entered from a terminal with EDIT=(YES,...) in the NAME macro to another terminal. It is not called when the message is inserted via a program-to-program switch.

The Message Switching (Input) edit routine does not support terminals that are defined dynamically using the Extended Terminal Option (ETO) feature.

Related Reading: For more information on ETO, see *IMS/ESA Administration Guide: Transaction Manager*.

Table 88 shows the attributes of the Message Switching (Input) edit exit routine.

Table 88. Message Switching (Input) Edit Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSCNTE0.
Link editing	Follow the guidelines described in “Link-Editing the Routines” on page 5.
Including the routine	No special steps are required to include this routine.
IMS callable services	To use IMS Callable Services with this routine, you must issue an initialization call (DFSCSII0) to obtain the Callable Service token and a parameter list in which to build the function-specific parameter list for the desired callable service. Use the ECB found in register 9 for the DFSCSII0 call. **Related Reading:**For details on using the initialization call, see “Step 2. Initializing IMS Callable Services (DFSCSII0)” on page 11. This exit routine is automatically linked to DFSCSI00 by IMS. No additional linking is required To use IMS Callable Services.
Sample routine location	IMS.TMSOURCE (member name DFSCNTE0).

Communicating with IMS

IMS uses the entry and exit registers to communicate with the routine.

Contents of Registers on Entry

Upon entry, the edit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	The buffer location of the input message segment after translation to EBCDIC and after IMS Basic Editing. The first two bytes of the buffer contain a binary message length. The third byte of the buffer is binary zeros. The binary count includes the 4-byte prefix. The fifth byte contains the first byte of message text.
7	Address of CTB.
9	Address of CLB.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of edit routine.

Use the message segment in the buffer addressed by register 1 as input to the edit routine.

The edit routine must place the text of the edited message segment to be returned to IMS in the buffer addressed by register 1. If the input was processed by the IMS Basic Edit, this buffer is always 10 bytes greater than the 2-byte binary count at the beginning of the message segment. The length of the message segment can be expanded or reduced to any desired size. The format of the edited message segment in the buffer upon return to IMS must be two bytes of binary count (LL), two bytes of binary zeros (ZZ), and edited text. The second two bytes (ZZ) should not be changed or edited. The LLZZ field is the first four bytes of the message segment.

Contents of Registers on Exit

Before returning to IMS, the edit routine must restore all registers except register 15, which must contain one of the following return codes.

Return Code	Meaning
00	Segment is processed normally.
04	Segment is canceled.
08	Message is canceled and the terminal operator is notified.
12	Message is canceled and the user message identified by register 1 is sent to the terminal.

Register 1 contains the message number if register 15 contains a return code of 12; otherwise it is ignored. Any other value causes the message to be canceled and the terminal operator to be notified.

Using the Sample Message Switching Edit Routine (DFSCNTE0)

The edit routine can be used to identify, in the text of the message to the output terminal, the logical terminal name from which the message was entered and the message number.

The following is an example of a Message Switching edit routine.

Assume the following message is being entered from a logical terminal named ‘XSYSI’ and is input message number one.

```
ABC SEND ALL XYZ MSGS TO THIS TERMINAL
```

The message received by the output terminal associated with logical terminal ABC has the input logical terminal name and input message number appended to it by the edit routine.

```
ABC SEND ALL XYZ MSGS TO THIS TERMINAL XSYSI 1
```

In this example, the input logical terminal name is used. This name is found in the Communication Name Table (CNT), which is the IMS control block for the input logical terminal. The CNT is addressed by a field called CTBCNTPT in the Communication Terminal Block. The field in the CNT containing the logical terminal name is called CNTNAME. Control blocks are defined in the *IMS/ESA Pseudo Module Listing*.

Chapter 46. MSC Link Receive Routing Exit Routines (DFSCMLR0/DFSCMLR1)

The Link Receive Routing exit routines can be used in combination with the Terminal Routing exit routine to reduce the number of remote transactions that must be defined.

Related Reading: The Terminal Routing exit routine is described in “Chapter 60. Terminal Routing Exit Routine (DFSCMTR0)” on page 403.

Message routing between IMS systems is based on logical terminals (LTERMs) and transactions, just as it is in a single-IMS system environment. In a multiple system configuration, LTERMs and transactions can be a part of any of the systems in the configuration.

Routing of messages is automatic, because each system in the configuration knows, via system definition, which transactions and logical terminals are local to it, and which ones it can reference by message switches, program-to-program switches, or transactions.

In an IMS system that includes Multiple Systems Coupling (MSC), additional control of message routing can be accomplished with the Routing exit routines described here. These exit routines do not apply to ISC.

About This Routine

The Link Receive Routing exit routines can reroute messages to different local transactions with the same characteristics. These routines receive control from IMS when the first segment of an input message is received from a remote system and the destination of the message is a local transaction. The Link exit routines can examine the first segment of the message or the SPA and reroute the message to another local transaction with like attributes.

DFSCMLR0

The MSC Link Receive Routing exit routine (DFSCMLR0) is specified on the COMM macro as part of the OPTIONS= parameter. MSLEXIT specifies that the MSC Link Receive Routing exit routine is to be included in the generated system.

Related Reading: For information on coding the COMM macro, see the section on “Macros”, in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Using IMS Callable Services with This Routine

To use IMS Callable Services with this routine, you must issue an initialization call (DFSCSII0) to obtain the Callable Service token and a parameter list in which to build the function-specific parameter list for the desired callable service.

Related Reading: For details on using the initialization call, see “Step 2. Initializing IMS Callable Services (DFSCSII0)” on page 11.

Use the ECB found in Register 9 for IMS Callable Services. This exit is automatically linked to DFSCSI00 by IMS. No additional linking is required to use IMS Callable Services.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of the edited (MFS or Basic) first segment or SPA.
2	Address of an 8-byte area containing the transaction code (left justified and padded with blanks).
9	Address of ECB.
11	Address of SCD.
13	Address of save area. The exit routine must not change the first three words.
14	Return address.
15	Entry point of exit routine.

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers except for registers 1, 2, and 15, which must contain the following:

Register	Contents
1	If the return codes are 0, 4, or 8, the contents of register 1 are destroyed. If the return code in register 15 is 12, the low-order two bytes contain the message key.
2	Address of the destination transaction, right-padded with blanks by the routine.
15	One of the following return codes:

Return Code	Meaning
0	No transaction code change.
4	The area pointed to by register 2 contains an updated transaction code.
8	The transaction code was rejected by the MSC exit routine. Message DFS2175 is issued to the input terminal; the input message was canceled and removed from the system.
12	The transaction code was rejected by the MSC exit routine. Register 1 contains a key for a user-keyed message to be issued to the inputting LTERM. The input message was canceled and removed from the system.

If the new destination does not have the same attributes as the original transaction, the invalid request is logged with a type '64' record and an error message is sent to the local master terminal and the input terminal.

DFSCMLR1

An alternate entry point (DFSCMLR1) can be established in DFSCMLR0. If the entry point exists, it is given control when the first segment of an input message is received from a remote system (sent via direct routing) and when the destination is a local terminal.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
1	Address of the segment.
2	Address of an 8-byte field containing the destination name.
3	Address of an 8-byte field containing the originating LTERM name.
13	Address of save area. The first three words must not be changed.
14	Return address.
15	Entry point of exit routine.

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers except for registers 2 and 15, which must contain the following:

Register	Contents
2	Must point to the same 8-byte field as upon entry. The contents of this field may be changed to point to another destination name.
15	One of the following return codes:

Return Code	Meaning
0	Use existing destination name.
4	Use new destination name.

The contents of registers 1 and 3 may be destroyed.

Using the Sample MSC Link Receive Routing Sample Routine (DFSCMLR0)

For the latest version of DFSCMLR0 and DFSCMLR1, see the IMS.TMSOURCE library; member names are DFSCMLR0 and DFSCMLR1.

Chapter 47. MSC Program Routing Exit Routine (DFSCMPR0)

This chapter describes the MSC Program Routing exit routine. Message routing between IMS systems is based on logical terminals (LTERMs) and transactions, just as it is in a single-IMS system environment. In a multiple system configuration, LTERMs and transactions can be a part of any of the systems in the configuration.

About This Routine

Message routing between IMS systems is based on logical terminals (LTERMs) and transactions, just as in a single IMS system environment. In a multiple system configuration, LTERMs and transactions can be part of any of the system configurations.

Routing of messages is automatic because each system in the configuration knows (via system definition) which transactions and logical terminals are local to it, and which ones it can reference by message switches, program-to-program switches, or transactions.

In an IMS system that includes Multiple System Coupling (MSC), additional control of message routing can be accomplished with the routing exit routine described here. This exit routine does not apply to ISC and is not used for conversation programs.

The MSC Program Routing exit routine provides additional routing capability for a system processing a message that is entered from a terminal in a different IMS system. This routine can allow the routing of a message as follows:

- Route a response message to an LTERM in the originating system different from the inputting LTERM.
- Route a message to any other LTERM in the originating system or remote MSC system without having to define the LTERM in the system changing the destination.
- Route a message to a transaction in the originating system or remote MSC system without having to define the transaction in the system changing the destination.

You can use this exit routine to do the following:

- Lessen the local system's requirement for using unique names to define LTERMs and transactions.
- Allow you to have the identical name for master terminals throughout your MSC network.
- Migrate LTERMs or transactions to other systems without having to change the application program or IMS definition in the processing system. In other words, the application program can change the destination from local to remote by issuing a CHNG call.

Related Reading: For information on coding the COMM macro, see the section on "Macros", in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Table 89 on page 346 shows the attributes of the MSC Program Routing exit routine.

Table 89. MSC Program Routing Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSCMPR0.
Link editing	Follow the guidelines described in “Link-Editing the Routines” on page 5.
Including the routine	No special steps are required to include this routine.
IMS callable services	DFSCMPR0 can use callable storage services. To use callable services with this routine, you must issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list to build the function-specific parameter list for the desired callable service. For details on using the initialization call, see “Step 2. Initializing IMS Callable Services (DFSCSII0)” on page 11. Use the ECB found in Register 9 for callable services. This exit routine is automatically linked to DFSCSI00 by IMS. No additional linking is required to use callable services.
Sample routine location	IMS.TMSOURCE (member name DFSCMPR0).

Communicating with IMS

This exit routine is given control whenever a DL/I CHNG call is issued to an alternate modifiable PCB by an application program processing in a local or remote MSC system and when:

- The Multiple System Coupling (MSC) feature is SYSGENED.
- The application program is processing a nonconversational transaction.

The program can request either symbolic routing or routing to the originating system. In the latter case, if the exit routine is being called due to a CHNG call from an application program processing in a remote MSC system (that is, a system that is remote to the input (source) LTERM system), then the LTERM or TRANCODE specified by the CHNG call, does not need to be defined in the processing system.

Contents of Registers on Entry

Upon entry to the exit routine, all registers must be saved using the save area provided. The registers contain the following:

Register	**Contents**
0	A 4-byte originating system identifier (SYSID) if exit routine is called by a remote MSC system (a system that is remote to the input (source) system) or, complemented 4-byte SYSID if exit routine is being called in the local (source) LTERM system (that is, the transaction is being processed locally in the input (source) LTERM system.
1	Address of an 8-byte field containing the originating LTERM name or blanks.
2	Address of an 8-byte field containing the LTERM name or transaction code supplied by the CHNG call.
9	ECB address.
11	Address of SCD.
13	Address of save area. The first three words must not be changed.
14	Return address.
15	Entry point of exit routine.

Contents of Registers on Exit

Upon return to IMS, all registers must be restored except for registers 0, 1, and 15. If the return code is 10 or 14, register 0 is set to the SYSID of the remote system to which the message is being routed (if routing via SYSID), or register 0 is set to 0 and register 1 is set to the address of an 8-byte MSNAME of the remote system to which the message is being routed (if routing via MSNAME). Register 15 contains the return code.

Return Code	Meaning
00	Use the destination name supplied by the CHNG call to find the destination in the processing system.
04	Use the originating system ID (SYSID) to route the message back to the LTERM supplied by the CHNG call in the originating system.
08	Return A1 status code to the application program. The DL/I CHNG call is not processed.
0C	Use the originating system ID (SYSID) to route the message back to the transaction code (supplied by the CHNG call) in the originating system.
1C	If this is a CHNG call for a directed routing request, then override the MSNAME destination with the LTERM/TRANCODE name in the I/O area of the ISRT call.
10	Route the message to the LTERM name supplied by the CHNG call in the remote system represented by either: SYSID in register 0 (set by DFSCMPR0)^{1}. 8-byte MSNAME pointed to by the address in register 1. Register 0 is set to 0.^{2}
14	Route the message to the transaction code supplied by the CHNG call in the remote system represented by either: SYSID in register 0 (set by DFSCMPR0)^{1}. 8-byte MSNAME pointed to by the address in register 1. Register 0 is set to 0.^{2}
18	Process message on originating IMS system. This return code is only valid for transactions that are defined as local (not remote) to the originating system and are not defined as SERIAL=YES

Notes:

1. When routing the message to a remote system (RC = 10 or 14), IMS tests register 0 (SYSID register) to determine if the exit routine is requesting to route the message using the SYSID or MSNAME. If routing via SYSID, register 0 is the SYSID (which is non-zero).
2. If routing via the MSNAME (RC= 10 or 14), register 0 is 0. IMS uses the MSNAME pointed to by the address in register 1 to reroute the message. The address pointed to in register 1 must be supplied by the exit routine. The exit routine can use the 8 bytes of storage pointed to by register 1 upon entry to store the MSNAME IMS is to use for rerouting the message.

Error Conditions

If IMS detects an error, an A1 status code will be returned to the application program issuing the CHNG call.

Related Reading: For more information on the A1 status code, see *IMS/ESA Application Programming: Transaction Manager*.

Chapter 48. OTMA Destination Resolution Exit Routine (DFSYDRU0)

The OTMA Destination Resolution exit routine is provided to allow you to determine and change the final destination of IMS Open Transaction Manager Access (OTMA) output messages.

About This Routine

The following rules apply for this exit routine:

- This routine is optional.
- This routine is not called if the destination is an IMS scheduler message block (SMB) name.
- This routine cannot override the originating LTERM name or OTMA transaction pipe name.
- This routine can only set the final destination once.

  If output is routed from one OTMA client to another, that client cannot use its own Destination Resolution exit routine to set a different final destination.

An OTMA client should not use a transaction name as a transaction pipe name (or routing key) because of potential conflict with the SMB name.

In an IMS subsystem, several or many OTMA Destination Resolution exit routines are allowed.

Table 90 shows the attributes of the OTMA Destination Resolution exit routine.

Table 90. OTMA Destination Resolution Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	The OTMA Destination Resolution exit routine is named according to the following rules: • The name specified on the Client-bid call is used if the client is active. • The client descriptor is used if the client is not active. • The name DFSYDRU0 is used if there is no client descriptor.
Link editing	Follow the guidelines described in “Link-Editing the Routines” on page 5. The OTMA Destination Resolution exit routine must be reentrant. The OTMA Destination Resolution exit routine must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.RESLIB. This exit routine is optional.
Including the routine	No special steps are required to include this routine.
IMS callable services	This exit routine is not eligible to use IMS Callable Services.
Sample routine location	IMS.TMSOURCE (member name DFSYDRU0).

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the exit routine.

Contents of Registers at Entry

At entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
R1	Address of standard IMS user exit routine parameter list
R13	Save area address (points to a single SAVEAREA, not a SAVEAREA chain)
R14	Return address
R15	Entry point address

Table 91 lists the standard IMS user exit parameters.

Table 91. Standard IMS Exit Routine Parameter List

Offset	Length	Description
+0	4	Address of user exit parameter list version number
+4	4	Address of the Callable Services token
+8	4	Address of the work area
+12	4	Address of the OTMA Destination Resolution exit routine parameter list

The following are the contents of the OTMA Destination Resolution exit routine parameter list.

Offset	Contents	
+0	Name of the originating LTERM or OTMA transaction pipe	
+8	Destination name. If the destination is for OTMA and no Tpipe name is specified in the output area, this field is used as the name of the Tpipe to queue and deliver the output message.	
+16	Transaction name or program name	
+24	Flag Type	
	Flags	**Description**
	X'80'	An OTMA prefix exists.
	X'40'	The exit routine can override the client name.
+25	Synchronization level	
+26	Destination type	
	Flags	**Description**
	X'80'	Transaction pipe exists for the client.
	X'40'	LTERM exists in IMS (Legacy).
	X'20'	LU 6.2 descriptor exists.
	X'10'	ETO is available.
	X'08'	Client is active.
+27	Reserved	
+28	User ID	
+36	Group name	
+44	Name of the destination OTMA client	
+60	Reserved	
+64	Name of the originating OTMA client, if the message originated from an OTMA client; otherwise blanks	
+80	Address of the input Message Control Information prefix section of the OTMA message	

Offset	Contents
+84	Address of the input State Data prefix section of the OTMA message Check the prefix flag in the Message Control Information section to determine the specific type of State Data section specified.
+88	Address of the input User Data prefix section of the OTMA message The area is also used to return new or modified user data, up to a maximum of 1024 bytes.
+92	Reserved
+96	Address of the output parameter list This parameter list is used to return information to IMS. The following are the contents of the output parameter list:

Offset	Contents
+0	The 16 byte client override name, if any This field is used when the destination is a different OTMA client. A return code of 8 must also be set.
+16	Output flag If this flag is set, a synchronized transaction pipe needs to be created. This flag can only be set if the return code is 0.
+17	Reserved (3 bytes)
+20	The 8-byte Tpipe name, if any. This field specifies the name of the Tpipe that is used for queuing and transmitting the output message. If it contains all blanks, the destination name is used for the Tpipe name. (This is only valid when return code is 0.)

Contents of Registers at Exit

Before returning to IMS, the exit routine must restore all registers, except register 15, which contains the return code.

Register	Contents
15	One of the following return codes:

Return Code	Meaning
0	Destination is the OTMA client transaction pipe
4	Destination is a non-OTMA LTERM name
8	Destination is a different OTMA client The new client name must be stored in the DFSYDRU0 parameter list.
12	Destination is invalid

Using the Sample OTMA Destination Resolution Exit Routine

A sample exit routine is provided. See the IMS.TMSOURCE library; the member name is DFSDRU0. For more information on this exit routine, see *MQSeries for MVS/ESA System Management Guide*.

Chapter 49. OTMA Input/Output Edit Exit Routine (DFSYIOE0)

The OTMA Input/Output Edit exit routine is provided to allow you to modify or cancel IMS Open Transaction Manager Access (OTMA) input and output messages.

About This Routine

This exit routine can do the following for OTMA input and output messages:

Modify the length or data of a message segment.

IMS sends the modified message once it receives control from the exit routine.

Cancel a message segment.

Table 92. Canceling a Message Segment

Segment Being Canceled	IMS Sends
First	The full OTMA message prefix, with null data.
Last	The last segment, with null data.
Other	Nothing. IMS does not send the message segment.

Cancel a message.

Segment Being Canceled	IMS Sends
First	Nothing. IMS does not send the message, and returns a status code.
Other	The last segment, with null data. In the OTMA prefix, the “discard chain” flag is set.

The length of each message segment is limited to 32K bytes. If a message segment exceeds this limit, IMS issues message `DFS1294E`, and processes the message as follows:

Segment Being Processed	IMS Sends
First	The full OTMA message prefix, with null data.
Last	The last segment, with null data.
Other	Nothing. IMS does not send the message segment.

Table 93 shows the attributes of the OTMA Input/Output edit exit routine.

Table 93. OTMA Input/Output Edit Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSYIOE0.

Table 93. OTMA Input/Output Edit Exit Routine Attributes (continued)

Attribute	Description
Link editing	Follow the guidelines described in “Link-Editing the Routines” on page 5. The OTMA Input/Output Edit exit routine must be reentrant. The OTMA Input/Output Edit exit routine must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.RESLIB. This exit routine is optional.
Including the routine	No special steps are required to include this routine.
IMS callable services	This exit routine is not eligible to use IMS Callable Services.
Sample routine location	IMS.SVSOURCE (member name DFSYIOE0).

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the exit routine.

Contents of Registers at Entry

At entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
R1	Address of standard IMS user exit routine parameter list
R13	Save area address
R14	Return address
R15	Entry point address

Table 94 lists the standard IMS user exit parameters.

Table 94. Standard Exit Parameter List (Mapped by DFSSXPL)

Field	Offset	Length	Description
SXPLVER	0	4	Address of word containing version number of standard exit parameter list.
SXPLATOK	4	4	Address of word containing callable services token. The callable services token must be included in the callable services parameter list when a callable service is requested.
SXPLAWRK	8	4	Static address of 512-byte work area for DFSAOE00. This is a permanent storage area allocated during IMS initialization and passed to DFSAOE00 each time it is called.
SXPLFSPL	12	4	Address of function-specific parameter list (Table 97 on page 358 shows this list).
SXPLINTX	16	4	Address of user data table loaded by DFSINTX0 at IMS initialization time. This field is valid in only IMS environments where DFSINTX0 is called. It will be zero in any other environment. For information about the DFSINTX0 exit routine, see “Chapter 36. Initialization Exit Routine (DFSINTX0)” on page 289.

Function-Specific Parameter List on Entry

The following are the contents of the OTMA Input/Output Edit exit routine parameter list.

Offset	Contents
+0	Input/output flag. Set to 0 for an input message segment; set to 4 for an output message segment.
+1	Segment-type flag. Set to 0 for the first message segment; set to 4 for any other message segment.
+2	Reserved.
+4	Address of the message segment The segment has the following format: LLZZDD: LL Total length (2 bytes) ZZ Flag (2 bytes) DD Message segment If the exit routine modifies the message segment, it must be sure to modify the LL with the new segment length. For null segments, LL should be set to 4 (2 bytes for LL and 2 bytes for ZZ). The exit routine can increase any segment to a maximum of 256 bytes. The overall message, however, cannot exceed 32 767 bytes (including the LL and ZZ fields). If a segment exceeds the 256 bytes limit, IMS truncates it and issues message DFS1967.
+8	Address of the transaction code.
+12	Address of the OTMA transaction pipe name.
+16	Address of the XCF member name.
+20	Address of the user ID.
+24	Address of the OTMA user table, if any.
+28	Address of message control information, available from input/output message prefix. This is an entry parameter only.
+32	Address of state data, available from input/output message prefix. This is an entry parameter only.
+36	Address of user data, available from input/output message prefix. This area can be used to return modified user data, but the length of user data cannot be changed. The format of the user data is: 0-1 Length of the user data that follows (including the length field). This user exit cannot change the length of user data. 2 User data.
+40	Address of output parameter list, The output parameter list returns information to IMS and defined as follows: +00 8-byte LTERM override. This field is used to override the destination override specified in the state data. +08 8-byte name override. This field is used to override the map name specified in the state data. +16 Reserved.

Contents of Registers at Exit

Before returning to IMS, the exit routine must restore all registers, except register 15, which contains the return code.

Register	Contents
15	One of the following return codes:

Return Code	Meaning
0	Processing continues.
4	Discard the message segment.
8	Terminate processing for the transaction.
12	Destination is invalid. Status AX will be returned to the application program and a 67D0 log record will be issued indicating error return code X'24'.

IMS treats any return code other than 4 or 8 as if it were 0, and processing continues.

Chapter 50. OTMA Prerouting Exit Routine (DFSYPRX0)

The OTMA Prerouting exit routine is provided to allow you to determine and change the destination of IMS Open Transaction Manager Access (OTMA) output messages. The messages can be routed to an OTMA client or to IMS TM for processing, but this exit routine cannot determine the final destination for the message.

Related Reading: For more information, refer to "Chapter 48. OTMA Destination Resolution Exit Routine (DFSYDRU0)" on page 349.

The OTMA Prerouting exit routine (if it exists) is the first one called for OTMA destination determination.

About This Routine

The following rules apply for this exit routine:

- This routine is optional, and can be written so that IMS data is not prerouted.
- This routine cannot override the final destination name.
- If the destination name is an IMS scheduler message block (SMB) name, this routine cannot change it.
- Transaction output can be directed to an OTMA client, even if the transaction originated from a non-OTMA source.
- Transaction output can be directed to a non-OTMA destination, even if the transaction originated from an OTMA client.

In an IMS subsystem, only one OTMA Prerouting exit routine is allowed.

Table 95 shows the attributes of the OTMA Prerouting exit routine.

Table 95. OTMA Prerouting Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSYPRX0.
Link editing	Follow the guidelines described in "Link-Editing the Routines" on page 5. The OTMA Prerouting exit routine must be reentrant. The OTMA Prerouting exit routine must be included in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.RESLIB. This exit routine is optional.
Including the routine	No special steps are required to include this routine.
IMS callable services	This exit routine is not eligible to use IMS Callable Services.
Sample routine location	IMS.SVSOURCE (member name DFSYPRX0).

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the exit routine.

Contents of Registers at Entry

At entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
R1	Address of standard IMS user exit routine parameter list
R13	Save area address (points to a single SAVEAREA, not a SAVEAREA chain)
R14	Return address
R15	Entry point address

Table 96 lists the standard IMS user exit parameters.

Table 96. Standard Exit Parameter List (Mapped by DFSSXPL)

Field	Offset	Length	Description
SXPLVER	0	4	Address of word containing version number of standard exit parameter list.
SXPLATOK	4	4	Address of word containing callable services token. The callable services token must be included in the callable services parameter list when a callable service is requested.
SXPLAWRK	8	4	Static address of 512-byte work area for DFSAOE00. This is a permanent storage area allocated during IMS initialization and passed to DFSAOE00 each time it is called.
SXPLFSPL	12	4	Address of function-specific parameter list (Table 92 on page 353 shows this list).
SXPLINTX	16	4	Address of user data table loaded by DFSINTX0 at IMS initialization time. This field is valid in only IMS environments where DFSINTX0 is called. It will be zero in any other environment. For information about the DFSINTX0 exit routine, see “Chapter 36. Initialization Exit Routine (DFSINTX0)” on page 289.

Function Specific Parameter List on Entry

The following are the contents of the OTMA Prerouting exit routine parameter list.

Table 97. Contents of the OTMA Prerouting Exit Routine Parameter List

Offset	Contents
+0	Name of the originating LTERM or OTMA transaction pipe
+8	Destination name
+16	Transaction name or program name
+24	Flags • If the first flag (+24) is set, an input OTMA prefix exists.
+25	Synchronization level
+26	Reserved
+28	User ID
+36	Group name
+44	Reserved
+48	Name of the originating OTMA client, if the message originated from an OTMA client; otherwise blanks
+64	Address of the input Message Control Information prefix section of the OTMA message

Table 97. Contents of the OTMA Prerouting Exit Routine Parameter List (continued)

Offset	Contents
+68	Address of the input State Data prefix section of the OTMA message Check the prefix flag in the Message Control Information section to determine the specific type of State Data section specified.
+72	Address of the input User Data prefix section of the OTMA message
+76	Reserved
+80	Address of the 16 byte client override name, if any, to be returned to IMS This field is set by IMS at entry. It points to a 16-byte buffer area to which the OTMA client name should be written, if one did not exist at entry. Do not alter this address. The OTMA client name is written when the transaction originates from a non-OTMA LTERM and is to be routed to an OTMA destination.

Contents of Registers at Exit

Before returning to IMS, the exit routine must restore all registers, except register 15, which contains the return code.

Register	Contents
15	One of the following return codes:

Return Code	Meaning
0	Route output to current destination. Route OTMA output to an OTMA destinations, and route non-OTMA output to non-OTMA destinations.
4	OTMA should process the message. If the transaction was entered from a non-OTMA LTERM, the OTMA client name must be provided in the OTMA Prerouting exit routine parameter list.
8	IMS (not OTMA) should process the message. The transaction was invoked from an OTMA TPIPE, but the destination is non-OTMA.

Using the Sample OTMA Prerouting Exit Routine

A sample exit routine is provided. See the IMS.SVSOURCE library; the member name is DFSYPRX0. For more information on this exit routine, see *MQSeries for MVS/ESA System Management Guide*.

Chapter 51. Output Creation Exit Routine (DFSINSX0)

The Output Creation exit routine allows you to validate an unknown destination for a message. IMS calls the Output Creation exit routine under the following conditions:

- The Extended Terminal Option (ETO) feature is active (ETO=Y), and an LTERM that is needed to queue a message for a user does not exist. The exit routine is called to create LTERMs for terminals for which ETO is active. IMS creates the LTERM from information in the user descriptor or from information that the Output Creation exit routine supplies. (For more information on ETO, see the *IMS/ESA Administration Guide: Transaction Manager*.)
- Shared queues are active (SHAREDQ=*name*) and the destination for the message cannot be found on the local IMS subsystem. The exit routine identifies the message destination as a transaction, LTERM, or an invalid destination. (To identify the message as an LTERM, ETO must also be active, and all of the rules that apply to the exit routine for ETO also apply for LTERMs identified for shared queues.) For valid transactions or LTERMS, IMS creates a resource block from information provided in the exit routine. Refer to “Using the Exit Routine with Shared Queues” on page 364 for more information on using the exit routine with shared queues.

About This Routine

When ETO=Y, you can write the Output Creation exit routine to perform the following tasks:

- Supply queue data to create LTERMs. This data can override the queue data that the user descriptor provides. If the user descriptor chosen is DFSUSER, the exit routine can also supply queue data to add additional LTERMs to the structure.
- Provide the correct user ID for the user receiving messages, overriding the one derived from the LTERM name. This name is also used as the name of the user control block structure. If no user ID is supplied, the name of the user structure is the name of the target LTERM (the LTERM to which the message is destined).
- Supply autologon parameters such as LU name, user ID, logon descriptor name, and mode table name. IMS automatically allocates the user to the indicated node and attempts to establish a session with that node.

Related Reading: For more information, see “Providing Queue Data and Autologon Parameters” on page 363.

The Output Creation exit routine is optional, unless you provide the Sign-On exit routine (DFSSGNX0, described in “Chapter 58. Sign-On Exit Routine (DFSSGNX0)” on page 391) because these two exit routines are corequisite. If you provide one exit routine to supply queue data for additional LTERMs, you must provide the other exit routine also. They both create the user control block structure and related LTERMs (including multiple LTERMs for a user): DFSINSX0 using an LTERM name and DFSSGNX0 using the user ID. Both exit routines must contain the same logic so that the user structure is identical, regardless of which exit routine created it.

When SHAREDQ=*name*, you can write the exit routine to perform the following tasks:

- Supply a destination type for an output message when the shared queues feature is active. The destination can be a transaction or an LTERM. IMS places the message on the appropriate shared queue.

- Supply transaction data to create transactions, such as whether the transaction is in conversational or response mode.

Table 98 shows the attributes of the Output Creation exit routine.

Table 98. Output Creation Exit Routine Attributes

Attribute	Description
IMS environments	TM environments.
Naming convention	You must name this exit routine DFSINSX0.
Link editing	Follow the guidelines described in "Link-Editing the Routines" on page 5. This exit routine must be reentrant. The exit routine can be called in cross-memory mode.
Including the routine	If you want IMS to call the Output Creation exit routine, include it in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.RESLIB. If the exit routine is included, IMS automatically loads it each time IMS is initialized if ETO=Y (after the Initialization exit routine, DFSINTX0, has changed the ETO= parameter) or if SHAREDQ=*name*.
IMS callable services	To use IMS Callable Services with this routine, you must do the following: • Issue an initialization call (DFSCSII0) to obtain the Callable Service token and a parameter list in which to build the function-specific parameter list for the desired callable service. For details on using the initialization call, see "Step 2. Initializing IMS Callable Services (DFSCSII0)" on page 11. • Use the ECB found at offset 0 of the Output Creation exit routine parameter list for the DFSCSII0 call. • Link DFSCSI00 with your user exit.
Sample routine location	IMS.TMSOURCE (member name DFSINSX0).

Restrictions

The Output Creation exit routine is not called during XRF tracking on an XRF alternate system.

Transaction control blocks created by the exit routine are not available for you to use in programming. These control blocks are used by IMS to place the transaction input message on the appropriate shared queue.

LTERM Processing for ETO

The following sections describe how the exit routine processing message for LTERMs when ETO=Y. See "Using the Exit Routine with Shared Queues" on page 364 for information on transaction and LTERM processing for shared queues.

Identifying Which User Descriptor IMS Selected

If the user control block structure already exists for the user for whom messages need queueing but for which the target LTERM is missing, IMS selects the user descriptor that was used to build the user structure and calls the exit routine.

Unlike the Sign-On exit routine, the Output Creation exit routine cannot return the address of a user descriptor. If no user control block structure exists, IMS searches for a descriptor that has the target LTERM defined. If IMS locates the target LTERM name, it selects that user descriptor and calls the Output Creation exit routine.

If IMS does not find a descriptor that contains the target LTERM name, it selects DFSUSER to create the user structure. IMS renames the descriptor, giving it the name of the target LTERM, and equates the user ID to this name. IMS then calls the Output Creation exit routine, which can supply the **correct** user ID, overriding the one derived from the target LTERM. If no user descriptor can be found, including DFSUSER, IMS rejects the LTERM creation request.

Providing Queue Data and Autologon Parameters

Depending on the user descriptor selected, the Output Creation exit routine can provide queue data (LTERM data) and autologon parameters. If the exit routine returns data that it is not allowed to return (as discussed in the cases following Figure 20), IMS rejects the LTERM creation attempt.

Figure 20 identifies two cases which describe what data the Output Creation exit routine can supply. The two cases are based on whether a DFSUSER or a non-DFSUSER descriptor is selected. (For this exit routine, non-DFSUSER descriptors are descriptors based on the target LTERM name.) Each case is discussed in the sections that follow. (For details on how your exit routine can supply this information, see “Contents of Registers on Exit” on page 366.)

If the Output Creation exit routine does not provide data to override the existing queue data, IMS proceeds as if you did not include the Output Creation exit routine; IMS uses the information in the selected user descriptor to create the LTERMs.

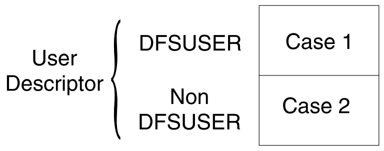


Figure 20. Queue Data and Parameters DFSINSX0 Can Provide

Case 1: If the DFSUSER descriptor is selected, the Output Creation exit routine:

- Can supply any of the fields defined in the interface (including LTERM names). The exit routine can change LTERM data, but not the actual name of the first LTERM provided.
- Can provide data for additional LTERMs.
- Can provide the correct user ID to override the user ID derived from the target LTERM.
- Can override autologon parameters. If the user structure already exists, the user’s existing autologon parameters are not changed.

IMS verifies the additional LTERMs that are specified against the LTERMs that already exist in the system. If an LTERM that is specified as an additional LTERM already exists in the system, IMS assumes that this LTERM has been assigned to a different user, and it is not made part of the user structure of the user for which messages are queued.

Case 2: If a non-DFSUSER descriptor is selected, the Output Creation exit routine can only specify queue data to override data derived from the user descriptor. The exit routine:

- Can supply queue data (except LTERM names) to override data that the descriptor provides,

- Can override autologon parameters. If the user structure already exists, the user's existing autologon parameters are not changed.
- **Cannot** provide data for additional LTERMs or override the user ID.

IMS verifies the LTERMs specified in the descriptor against the LTERMs that already exist in the system. If an LTERM that is specified in the descriptor already exists in the system, IMS assumes that this LTERM has been assigned to a different user, and it is not made part of the user structure of the user for which messages are queued.

Using the Exit Routine with Shared Queues

IMS calls the exit routine if shared queues are active and the destination for the message is not found on the local subsystem. The exit routine determines the destination type for the message.

IMS uses the information returned by the exit routine to create a resource block for the destination. The resource control block can be an LTERM, or a transaction control block.

When the exit routine indicates that the destination is a transaction, IMS creates a transaction control block. DFSINSX0 returns information to IMS about the transaction, including whether the transaction is in conversational or response mode, and the SPA size if applicable. The transaction control block is not deleted until IMS restart. IMS can use the same transaction control block if it encounters additional instances of the undefined transaction input message.

Restriction: You cannot use the transaction control block for any purpose, such as scheduling the transaction to run in the local IMS subsystem. The transaction control block is only used by IMS to enqueue the input message on the appropriate shared queue.

Resource Control Block Creation

The resource control blocks created by the output creation process are shown in Table 99.

Table 99. Resource Blocks Created by the Output Creation Exit Routine

ETO Active	Shared Queues Active	DFSINSX0 Included	Force Create (IOPCB)	ETO Descriptor Found [1]	Exit Routine Action
No	No				No control block created.
No	Yes	No			No control block created.
No	Yes	Yes	No		Transaction control block created.
No	Yes	Yes	Yes		No control block created.
Yes	No			No	No control block created.
Yes	No			Yes	LTERM created.
Yes	Yes	No		No	No control block created.

Table 99. Resource Blocks Created by the Output Creation Exit Routine (continued)

ETO Active	Shared Queues Active	DFSINSX0 Included	Force Create (IOPCB)	ETO Descriptor Found [1]	Exit Routine Action
Yes	Yes	No		Yes	LTERM created.
Yes	Yes	Yes	No	No	Transaction control block created.
Yes	Yes	Yes	No	Yes (DFSUSER only)	LTERM or transaction control block created.
Yes	Yes	Yes	No	Yes (non-DFSUSER)	LTERM created.
Yes	Yes	Yes	Yes	No	No control block created.
Yes	Yes	Yes	Yes	Yes	LTERM created.

Note [1]: Unless otherwise noted, the ETO descriptor can be any type of ETO user descriptor.

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the exit routine.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
R1	Address of Standard Exit Parameter List (see Table 100)
R13	Save area address
R14	Return address to IMS
R15	Entry point address of exit routine

Table 100. Standard Exit Parameter List (Version 1)

Offset	Length	Description
+0	4	Address of standard exit parameter list version number
+4	4	Reserved
+8	4	Reserved
+12	4	Address of Output Creation exit parameter list
+16	4	Reserved

Table 101 shows the Output Creation exit routine parameters.

Table 101. Output Creation Exit Parameter List

Offset	Length	Description
+0	4	ECB address.
+4	4	SCD address.
+8	4	User Table address.
+12	4	Address of a buffer for use by the exit routine to return user ID and queue data. For additional details on the content and format, refer to the prolog in the sample routine (DFSINSX0 in IMS.TMSOURCE). The value is zero for the following conditions: • An XRF alternate system. • The destination must be a transaction and there is an indicator at offset +20. See Figure 99 for the conditions that indicate a transaction destination type.
+16	4	Address of a buffer for use by the exit routine to return Autologon Override parameters. For additional details on the content and format, refer to the prolog in the sample routine (DFSINSX0 in IMS.TMSOURCE). The value is zero for the following conditions: • An XRF alternate system. • The destination must be a transaction and there is an indicator at offset +20. See Figure 99 for the conditions that indicate a transaction destination type.
+20	4	Address of a buffer containing destination name, and other environment flags, including indicators for the following: • ETO or shared queues is enabled. • LTERMs or transaction control blocks can be created. • The exit routine output is an LTERM or a transaction control block.
+24	4	Address of a buffer for use by the exit routine. The buffer returns information that is used to create a transaction control block if the destination is a transaction. The value is zero if the destination is an LTERM. See Figure 99 for the conditions that indicate a transaction destination type.

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers except for registers 15, which contains the return code.

Register	Contents
15	One of the following return codes. If an application INSERT call forced the LTERM creation, IMS ignores the return code.

Return Code **Meaning**

Register	Contents	
	0	IMS creates the destination.
	¬0	IMS rejects the destination creation attempt for alternate PCBs. If an application INSERT call caused IMS to attempt the destination creation, the ¬0 return code is returned to the application as an 'A1' status code. IOPCBs force LTERM creation and ignore the return code.

In addition to the return codes, the exit routine contains an indicator at offset +20 with the value LTERM or TRAN to indicate whether an LTERM or a transaction control block was created. LTERM is the default (unless LTERM creation is not allowed).

Chapter 52. Physical Terminal (Input) Edit Routine (DFSPIXT0)

This chapter describes the Physical Terminal (Input) Edit routine. This user-written edit routine gains control before the IMS Basic Edit routine. If the input message is processed by MFS, the Physical Terminal (Input) edit routine is not called. This edit routine is called only when inserted from a terminal; it is not called when the message is inserted via a program-to-program switch. This edit routine is not called for LU 6.2 terminal input.

About This Routine

Message segments are passed one at a time to the Physical Terminal Input edit routine, and the edit routine can handle them in one of the following ways:

- Accept the segment and release it for further editing by the IMS basic edit routine.
- Modify the segment and release it for further editing by the IMS basic edit routine. Examples of segment modifications that can be made are changing the transaction code and reformatting the message text. Make any required modifications, because IMS has not yet performed destination or security checking.
- Cancel the segment.
- Cancel the message and request that the terminal operator be notified accordingly.
- Cancel the message and request that a specific message from the User Message Table be sent to the terminal operator.

The Physical Terminal Input edit routine requests the above actions by specifying different return codes that are interpreted and acted upon by IMS.

Bypassing Basic Edit

If the IMS application program supplies DFS.EDTN in the MOD name parameter for the output message, the IMS basic edit routine will be bypassed except for transaction code and password validation.

Related Reading: For further information, see “MFS Bypass for the 3270 or SLU 2” in the “Application Programming Using MFS” chapter in *IMS/ESA Application Programming: Transaction Manager*.

The Physical Terminal Input edit routine must position the transaction code, and optionally the password, if the terminal is not operating in conversational or preset destination mode. The exit routine should detect errors and return a message to the terminal operator if any errors are found.

IMS maintains a flag in the CTB (bit CTB6TRNI in the CTBFLAG6 field) to indicate when 3270 MFS bypass, nonconversational, no preset destination, and first segment exist upon input to the Physical Terminal Input exit routine. This flag notifies the Physical Terminal Input exit routine that it can add a minimum of 1 and a maximum of 18 bytes to the front of the message segment for a transaction code and optional password. The minimum of 1 byte to be added to the front of the message segment consists of a 1-byte transaction code. If NOBLANK is not specified at system generation, a minimum of 2 bytes is added to the front of the message segment, consisting of a 1-byte transaction code and 1 blank, which is necessary as a separator. To add a transaction code and optional password, the exit routine can put a return code of 16 in register 15 and set register 1 to point to an LLZZ field, followed by the data to be added.

Specifying the Routine

The Physical Terminal Input exit routine (DFSPIXT0) is specified on the LINEGRP or TYPE macros as part of the EDIT= parameter. If you are using **both** the Physical Terminal Input and Output edit routines, you must specify (YES,YES) on the EDIT= parameter of the TERMINAL macro or Extended Terminal Option (ETO) logon descriptor.

The CSECT name for this edit routine is the name specified in the TYPE or LINEGRP macro statement for which this edit routine applies. You must also specify YES in the EDIT= parameter of the TERMINAL macro statement or ETO logon descriptor.

The Global Physical Terminal Input edit routine (DFSGPIX0) performs the same functions as this edit routine but does not require system definition.

Related Reading:

- For information on coding the LINEGRP, TYPE, and TERMINAL macros, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.
- For more information on the ETO feature, see *IMS/ESA Administration Guide: Transaction Manager*.
- For more information on this edit routine, see “Chapter 34. Global Physical Terminal (Input) Edit Routine (DFSGPIX0)” on page 283.

Using IMS Callable Services with This Routine

To use IMS Callable Services with this routine, you must issue an initialization call (DFSCSII0) to obtain the Callable Service token and a parameter list in which to build the function-specific parameter list for the desired callable service. Use the ECB found in register 9 for the DFSCSII0 call.

Related Reading: For details on using the initialization call, see “Step 2. Initializing IMS Callable Services (DFSCSII0)” on page 11.

This exit is automatically linked to DFSCSI00 by IMS. No additional linking is required to use IMS Callable Services.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of Registers on Entry

Upon entry to the edit routine, all registers must be saved using the save area provided. The registers contain the following:

Register	Contents
1	Address of the input message segment buffer. IMS editing has not been performed. The first two bytes of the buffer contain the segment length (binary length includes the 4-byte overhead). The third and fourth bytes of the buffer are binary zeros. The message text begins in the fifth byte of the buffer. If the device was defined with MFS support but this message is not being processed by MFS, the first segment of the message has backspace error correction performed before entry to this edit routine. If escape (**) was entered by the terminal operator, the first two data bytes have been changed to binary zeros.
7	Address of CTB for the physical terminal from which the message was entered.
9	Address of CLB for the physical terminal from which the message was entered.
13	Address of save area. The first three words must not be changed.
14	Return address to IMS.
15	Entry point of edit routine.

The edit routine you supply can edit the message segment in the buffer pointed to by register 1.

You can reduce the length of the message segment to any size by replacing the length in the buffer with the appropriate value. The length field must appear in the same place at exit as at entry, and bytes 3 and 4 must not be changed.

Contents of Registers on Exit

Upon return to IMS, all registers must be restored except for register 1, which contains a message number if register 15 contains a value of 12; otherwise it is ignored. Register 15 contains one of the following return codes:

Return Code	Meaning
00	Segment is processed normally.
04	Segment is canceled.
08	Message is canceled and the terminal operator is notified.
12	Message is canceled, and the message identified by register 1 is sent to the terminal.
16	Insert the transaction code and optional password following the LLZZ pointed to by register 1. This return code is only valid for 3270 MFS bypass terminals. When the entering terminal is not a 3270 MFS bypass terminal, and the physical terminal input exit gives a return code of 16, IMS issues an error message, and the transaction code is not inserted in the message.

Any other return code causes the message to be canceled and the terminal operator to be notified.

Using the Sample Physical Terminal Input Edit Routine (DFSPIXT0)

This routine performs the following functions:

- Scans the input message segment for an expected format (TESTEXIT)
- Generates return codes (XX) based on the input request (TESTEXIT,XX)
- Verifies the user message number (YYY) if specified (TESTEXIT,XX,YYY)

- Replaces TESTEXIT with ERROR if return code or message number is invalid and passes the segment to IMS (return code 0)

For the latest version of the DFSPIXT0 source code, see the IMS.TMSOURCE library; member name DFSPIXT0.

Chapter 53. Physical Terminal (Output) Edit Routine (DFSCTTO0)

This chapter describes the Physical Terminal (Output) edit routine.

About This Routine

The Physical Terminal (Output) edit routine enables you to edit output messages immediately before they are sent to a terminal. During system definition, you specify which physical terminals or set of VTAM nodes use the defined edit routine for output editing. You can specify one Physical Terminal (Output) edit routine for each BTAM telecommunication line group. You can use these edit routines to meet your special editing needs associated with different communication terminals.

An output message can be processed by 1) a Physical Terminal Output edit routine and the IMS Basic Edit routine or 2) a Physical Terminal Output edit routine and MFS (Message Format Service). Output editing is performed in this sequence. Therefore, the input to the edit routine is the output of the application program, and the output of the edit routine is the input to MFS or the IMS Basic Edit routine.

You can also specify that this edit routine cancel an output message so that it is not delivered to the terminal. Instead, the routine can optionally request that an error message be sent in place of the canceled message.

The following criteria apply to message cancelation:

- Output messages can be canceled if they are destined for VTAM terminals only. Requests to cancel a message to a BTAM terminal are ignored by IMS (for example, if an LTERM is reassigned from a VTAM to a BTAM terminal, IMS ignores requests to cancel an output message).
- Conversational output and IMS in-core system messages cannot be canceled. Such cancelation requests from the exit are ignored, and the output message is sent.
- The request to cancel must be made for the first segment of a message. Requests for non-first segments of a message to be canceled are ignored, causing normal output processing to continue for the message.
- This routine is not activated for messages going across an MSC VTAM link, so these messages cannot be canceled.

Specifying the Routine

The Physical Terminal Output edit routine (DFSCTTO0) is specified on the LINEGRP or TYPE macro as part of the EDIT= parameter. If you are using **both** the Physical Terminal Input and Output edit routines, you must specify (YES,YES) on the EDIT= parameter of the TERMINAL macro.

Related Reading: For information on coding the LINEGRP, TYPE, and TERMINAL macros, see the section on “Macros” in *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Using IMS Callable Services with This Routine

To use IMS Callable Services with this routine, you must issue an initialization call (DFSCSII0) to obtain the Callable Service token and a parameter list in which to build the function-specific parameter list for the desired callable service. Use the ECB found in register 9 for the DFSCSII0 call.

Related Reading: For details on using the initialization call, see “Step 2. Initializing IMS Callable Services (DFSCSII0)” on page 11.

This exit routine is automatically linked to DFSCSI00 by IMS. No additional linking is required to use IMS Callable Services.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of Registers on Entry

Upon entry, the edit routine must save all registers using the provided save area. The registers contain the following:

Register	**Contents**
1	The address of a buffer containing the output message segment to be edited. The first two bytes are a binary count of the message segment length. The second two bytes are control information provided by the application program that constructed the message. The text of the output message starts in byte 5. The count includes the first four bytes in length. This register contains zeros if flag CTBAEOM in field CTBACTL of the CTB is on, indicating end-of-message. Any exit that modifies the contents of the buffer passed in register 1 should test for an end-of-message condition.
2	The address of an 8-byte field that contains either binary zeros or the user ID associated with the output message. The contents of the user ID field are described in the *IMS/ESA Application Programming: Design Guide* in the section on “I/O PCB Masks” in Chapter 6, “Designing MFS Applications”. The user ID in the output message can be compared to the user ID in the CTB (CTBUSID) to determine editing requirements. The user ID is only checked on the first segment of a multisegment message. DFSCTTO0 uses CTBAEOM and ENTSTAT to determine which segment is being processed.
7	CTB address for the destination terminal. **CTBFLAGC field**: CTBCDSDT bit *on* means that session restart has occurred for this terminal. If the edit routine is called with the CTBCDSDT on, the edit can assume that this is the first application output message selected for output processing since the session has restarted (provided that the bit is turned **off** by the edit routine after the first message is processed). IMS turns this bit **on** every time SDT (Start Data Traffic) for VTAM occurs. The edit routine is responsible for resetting this flag after it receives the first message. **CTBFLAG4 field**: CTB4RESP bit **on** means that the terminal is in response mode. After a system restart, IMS resets CTB4RESP.
9	Address of CLB. This block starts with a DECB. The content of the DECAREA field in the DECB is equivalent to the content of register 1.
11	Address of SCD.
13	Address of save area. The edit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of edit routine.

The output message segment that your edit routine returns to IMS from must be pointed to by the contents of the DECAREA field of the DECB. The first four bytes must be in a format as received at input with the binary count updated to the edited message segment length inclusive of the four bytes of prefix.

Contents of Registers on Exit (If No Cancel Request)

Before returning to IMS, the edit routine must restore all registers. If you are editing the message in place, you can increase its length by a **maximum** of ten bytes.

When the last segment of a message has been edited, IMS returns control to the routine. The routine has no new message data to edit.

Whenever a Physical Terminal Output edit routine is called, the CTB is in register 7. A 1-byte field, CTBACTK, in the CTB contains a 1 in the second bit position if this entry to the routine is for end of message (EOM).

Contents of Registers on Exit (If Cancel Request)

Upon return, registers must be restored except for register 15, which must contain the following return code.

Register	Contents	
15	The following return code:	
	Return Code	**Meaning**
	4	Message canceled. The buffer length must be set to zero.

All registers are not restored when a cancel request is made and the edit requests that IMS send an error message DFS3489I to the terminal for a non-response-mode message.

In order for IMS to cancel an output message before it is sent to the terminal, the Physical Terminal Output edit routine must make a request when the first segment of an output message is presented to it. The edit makes this request by setting the length of the first segment to zero in the buffer pointed to by DECAREA.

If the edit routine wants IMS to send error message DFS3489I in place of the canceled message, it places a return code of 4 in register 15 (in addition to zeroing the length field of the first segment).

If the terminal is in response mode, IMS always replaces the canceled message with error message DFS3489I. Across a system restart, response mode is reset. Therefore, if an output message is canceled after the system restart, no error message is sent.

If the terminal is not in response mode, the edit routine is not required to have IMS send error message DFS3489I. Nevertheless, it may be necessary to have IMS send the error message to prevent a hang condition for certain device types that are expecting a message.

Related Reading: For an explanation of error message DFS3489I, see *IMS/ESA Messages and Codes*.

Using the Sample Physical Terminal Output Edit Routine (DFSCTTO0)

The sample example shows how to extend an output message and how to attach a prefix. IMS Callable Services are used to get and release storage. This example applies to single-segment or multisegment messages, and to as many devices as the edit routine's table is assembled to handle. The default table size allows for five devices, but can be changed by modifying the label NUMENTS. If the table capacity is exceeded, an ABENDU55 results. If the prefix had not increased the message length by more than ten bytes, it could have been attached without the creation of an additional buffer area. For the latest version of DFSCTTO0, see the IMS.TMSOURCE library; member name is DFSCTTO0.

Chapter 54. Queue Space Notification Exit Routine (DFSQSPC0)

The Queue Space Notification exit routine (DFSQSPC0) is activated when a logical record is assigned to or released from a message queue data set. This routine causes a message to be issued when one of following occurs:

- The number of records currently in use exceeds upper threshold percentage value of the maximum number assignable before initiation of automatic shutdown.
- The number of records currently in use falls below the lower threshold percentage value of the same maximum.

In a shared queues environment the common queue server (CQS) returns the following information when this exit routine runs.

- The shared queue structure is in an overflow state.
- The destination queue in a shared queues environment is in an overflow state.

IMS sets an upper threshold value of 75 percent, and a lower threshold value of 60 percent. You can modify these values using the QTU and QTL parameters of the IMS procedure.

Related Reading: For details on these parameters, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

QTU has a range of 2 percent through 100 percent, and QTL has a range of 1 percent through 99 percent.

About This Routine

The SHUTDWN parameter of the MSGQUEUE macro allows you to reserve a number of records in each message queue data set. If the data set fills up with unprocessed messages, one of the following occurs:

- The system automatically shuts down with an internal CHECKPOINT DUMPQ.
- In a shared queues environment the system automatically shuts down with a CHECKPOINT FREEZE.

If unprocessed messages overflow a message queue data set before the automatic shutdown completes, a U0758 abend occurs.

This exit routine provides a warning before the automatic shutdown is initiated, so you can reduce message queue buildup, possibly avoiding the automatic shutdown and, most importantly, the U0758 abend.

You can replace the IMS-supplied exit routine with your own to establish your own threshold algorithm and/or issue user messages, which can then be captured by the AOI exit routine to reduce queue usage.

As an option, for certain units of work, you can modify this exit to find the number of records currently in use by the calling task. You can also request information that can be used to terminate the unit of work. For each application and each LU 6.2 conversation, IMS maintains counts of short and long message queue records (DRRNs) assigned, and supplies them to DFSQSPC0 if this option is used.

If you use this option, the expanded parameter list contains an output field that allows you to tell IMS that you want the unit of work stopped because one or both of the counts have exceeded specified limits. Different count limits can be established for different tasks.

For most program types queue counts are reset when one of the following occurs:

- A message is retrieved (GU call) from the message queues.
- A sync point occurs.
- A rollback occurs.
- The application terminates normally.

Exception: For non-message-driven BMPs, and for multiple-mode transactions (MODE=MULT specified on the TRANSACT macro), the queue counts are not reset until normal termination. If the unit of work is a DC transaction or conversation the counts are not provided.

The exit routine terminates a unit of work in the following ways:

- For an application program, an 'A7' status code is returned to the application. If the AIBTDLI call interface was used, the application also gets an AIB return code (X'104') and a reason code (X'190'). If the application tries to insert a message after the unit of work terminates, it immediately receives an 'A7' status code and the call is not processed.
- For an LU 6.2 conversation, the conversation is deallocated.

After the unit of work terminates, message queue records in use are released.

The sample exit routine DFSQSPC0 (IMS.TMSOURCE) describes how to enable this option. Using this option creates some additional overhead from building the expanded parameter list. The default for this option is NO.

Table 102 shows the attributes of the Queue Space Notification exit routine.

Table 102. Queue Space Notification Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSQSPC0.

Link editing	Follow the guidelines described in “Link-Editing the Routines” on page 5.
	
	This routine must be reentrant. It can be called in cross-memory mode.
	
	DFSCSI00 (Callable Services module) must be included in this load module if you plan to use IMS Callable Services from this exit routine. An example of the link-edit control statements is:
	`INCLUDE LOAD(DFSQSPC0)     SPACE NOTIFY USER EXIT`
	`INCLUDE LOAD(DFSCSI00)     IMS CALLABLE SERVICES`
	`MODE    AMODE(31),RMODE(ANY)`
	`ENTRY   DFSQSPC0`
	`NAME    DFSQSPC0(R)`

Including the routine	DFSQSPC0 is a separately linked composite module in the IMS.RESLIB. If you write your own exit routine, it must be linked into IMS.RESLIB.

Table 102. Queue Space Notification Exit Routine Attributes (continued)

Attribute	Description
IMS callable services	To use callable services with this routine, you must issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. For details on using the initialization call, see “Step 2. Initializing IMS Callable Services (DFSCSII0)” on page 11. The IMS-provided version of DFSQSPC0 includes an example that uses callable services to obtain working storage during an initialization call to DFSQSPC0. Use the ECB passed in parameter list QSPCECB for IMS Callable Services.
Sample routine location	IMS.TMSOURCE (member name DFSQSPC0). The DFSPARM macro is in IMS.GENLIBB (member name DFSPARM).

Restrictions

The following restrictions apply to DFSQSPC0:

- MVS services are unavailable to programs running in cross-memory mode unless the service’s documentation specifically states that it is available.
- Code running in cross-memory mode cannot issue any SVCs except ABEND.

Because this exit is called every time a message queue data set record is assigned or released, the logic you add to this exit can have a negative effect on system performance. IWAITs, time consuming algorithms, and excessive use of IMS Callable Services should be avoided.

If you want to issue user messages instead of IMS system messages DFS2013 through DFS2018, you must provide an exit which returns user message keys in register 15. The value returned in register 15 is actually the negative of the key in the user message table. In addition to returning the appropriate message key in register 15, be sure the message text is in the user-supplied message table, DFSCMTU0.

Related Reading: See “Chapter 28. User Message Table (DFSCMTU0)” on page 213 for information on creating and including a table.

Exit Routine Call Types

The following call types are recognized by the Queue Space Notification exit routine. Some of the parameters passed to the exit routine vary with the call type.

Call Type	Description
1	Initialization call
2	Application assigned DRRN (for example, a DL/I application)
3	LU 6.2 assigned DRRN
4	Not supported
5	Free the DRRN
6	OTMA assigned DRRN

Communicating with IMS

The queue space notification exit routine is called whenever a logical record is assigned to or released from a message queue data set. A parameter list is passed to the exit routine. Its contents depend on whether the user-provided DFSQSPC0 takes advantage of the optional capabilities provided by IMS. The IMS-provided DFSQSPC0 does not use the optional capabilities, although it does describe how they can be used.

To take advantage of the optional capabilities, you must modify DFSQSPC0 to recognize the initialization call type (Type 1) and, when the call is made, turn on the bit QSPCF2IN in the parameter list field QSPCFLG2. IMS responds to this flag being set by setting a flag in the SCD. The SCD flag tells IMS to support the new capabilities and provide the expanded parameter list to DFSQSPC0. This call is made only during early IMS (Queue Manager) initialization to enable the user exit to obtain working storage that is always to be available to DFSQSPC0 via the parameter list.

User-provided versions of DFSQSPC0 need not change if the message record count capability is not used.

The parameter list is mapped by the macro DFSPARM. The parameter list has five sections:

1. Message queue data set in-use count and threshold status
   - The number of records currently in use

     The high-order byte of the in-use count is used as a flag byte.
   - The maximum number of records assignable before shutdown

     The exit routine interrogates these values and sets the parameter flag and a return code (register 15) based on their values. The return code is either zero or an error message number.
2. Pointers to control blocks and thresholds

   These fields are always passed to DFSQSPC0:
   - Address of the SCD control block
   - Address of the ECB (required for IMS Callable Services).
   - Address of user exit’s workarea or zero
     - During the initialization call to DFSQSPC0, you can use IMS Callable Services to obtain working storage for your exit. The address you store in the parameter list is saved by IMS and returned to your exit on every call. IMS only saves the address returned by the exit during the initialization call. Addresses returned during other calls are overlaid by the address returned from the initialization call, or by zero if no address was returned.
     - User-provided exit is responsible for obtaining the workarea during the initialization call to DFSQSPC0 and storing its address in the parameter list.
   - Upper and lower threshold limits (Same values as found in SCDQTU and SCDQTL)
3. Type of Call and other input/output flags

   The following fields are only used while processing Call Types: 1, 2, 3, 5, and 6. In all other cases, the Call Type is set to zero.
   - Call type
   - Assign/Free DRRN indicator

- Message Queue record count exceeded flag - set by exit

   The following flags can be set if shared queues are active:
   - Shared queue structure is in an overflow state.
   - Destination queue is in an overflow state.
4. Unit of Work Information

   These fields are only used while processing a DL/I (Call Type 2) application or an LU6.2 (Call Type 3) terminal request:
   - Accumulated counts of short and long message queue records assigned by this unit of work
   - Identification of the unit of work making the call:

     For a DL/I application: TRAN Name, PSB Name, and Terminal Symbolic

     For an LU6.2 Terminal: LU Name, TP Name and length, Side Name

     For an OTMA client: Tpipe name, XCF member name, and override LTERM name
5. Message destination name

   The message destination name is used while processing Call types 2, 3, 5, and 6. If the destination name is not available at the time of the call, this field is set to zero.

The IMS-supplied Queue Space Notification exit routine found in IMS.TMSOURCE can be used as a guide in creating your own exit routine.

If you want to change the threshold notification algorithm, note the following interface requirements.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents	
0	Data set indicator:	
	00	QBLKS data set
	04	SMSGQ data set
	08	LMSGQ data set
2	Address of parameter list described below	
9	Address of ECB	
10	Address of SCD	
14	Return address to IMS	
15	Entry point of exit routine	

Description of Parameters

The macro DFSPARM generates the DSECT for the parameter area passed to DFSQSPC0 by IMS. For additional information, refer to DFSPARM included in IMS.GENLIBB.

A pointer to the SCD is contained in the input field QSPCSCD as well as in Register 10.

Recommendation: It is recommended that you get addressability to the SCD from the parameter list in anticipation of a future release when Register 10 will no longer pass the SCD.

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers except for register 15, which must contain one of the following return codes:

Return Code	Meaning
0	No message issued
Message key	Queue Manager issues a message
Negative	User-defined message sent

Threshold Values

The parameter list fields QSPCQTU and QSPCQTL contain the upper and lower threshold values.

The thresholds are either:

- IMS defaults (75 percent and 60 percent).
- Your default values specified in QTU and QTL in the DFSPBxxx member.
- QTU and QTL values specified in the IMS procedure. (See “Chapter 1. Guidelines for Writing IMS Exit Routines” on page 3.)

Chapter 55. Security Reverification Exit Routine (DFSCTSE0)

This chapter describes the Security Reverification exit routine (DFSCTSE0). This exit routine allows you to re-evaluate transaction authorization checking on the DL/I CHNG Call. Transaction Manager applications that use Multiple System Coupling (MSC), CHNG calls, and AUTH calls on a remote IMS system can benefit from coding this exit routine. By coding this exit routine, you can avoid a security failure that occurs when RACF or a non-RACF security environment is called in a destination MSC system by a user that is not signed on to that particular IMS system.

Related Reading: This exit routine can be written as part of the Transaction Authorization exit routine. See "Chapter 62. Transaction Authorization Exit Routine (DFSCTRN0)" on page 411 for more information.

About This Routine

This exit routine is an entry point in DFSCTRN0. If you do not code this entry point, IMS does not call it.

Table 103 shows the attributes of the Security Reverification exit routine.

Table 103. Security Reverification Exit Routine Attributes

Attribute	Description
IMS environments	TM environments.
Naming convention	You must name this exit routine DFSCTSE0.
Link editing	DFSCTSE0 must be link edited to DFSCTRN0, or coded as an explicit part of DFSCTRN0. If you code this entry point, it should have access to a table of valid user IDs, passwords, and transactions associated with each valid user ID, or contain some algorithm to derive this authorization information. For addressability, this table should reside in this module, in the /SIGN ON exit (DFSCSGN0), or in the IMS nucleus.
Including the routine	If DFSCTSE0 is link edited to DFSCTRN0, it is called upon return from DFSCTRN0. See "Communicating with IMS" for more information on the way IMS calls DFSCTSE0.
IMS callable services	To use callable services with this routine, you must do the following: • Issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. For details on using the initialization call, see "Step 2. Initializing IMS Callable Services (DFSCSII0)" on page 11. • Use the ECB found in register 9 for the DFSCSII0 call. • Link DFSCSI00 with your user exit routine.
Sample routine location	IMS.TMSOURCE (included as part of member name DFSCTRN0).

Communicating with IMS

You can call the exit routine in either a RACF environment or a non-RACF environment.

If you are operating in a RACF environment, you can activate this routine in the following place:

- If the RACF FRACHECK call returns a valid return code (0 or 4), IMS immediately calls the exit routine DFSCTRN0 if DFSCTRN0 exists. Upon returning from DFSCTRN0, regardless of its return code, IMS will immediately call DFSCTSE0 if DFSCTSE0 exists. This applies to CHNG calls only.

If you are operating in a non-RACF environment, you must activate this routine in the following place:

- If DFSCTSE0 is coded as an entry point, IMS calls this entry point following each call to DFSCTRN0 if DFSCTRN0 exists. This applies to CHNG calls only. IMS calls this entry point regardless of the return code received from DFSCTRN0.

Whether you are operating in a RACF or non-RACF environment, DFSCTRN0 passes the return code directly to DFSCTSE0 in register 3.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the save area provided. The registers contain the following:

Register	Contents
0	Address of the user ID from PST (PSTUSID)
1	Address of the password or zero For AUTH call, address of GENERIC class For TRAN call, address of TRAN class For FIELD call, address of FIELD class For DATABASE call, address of DATABASE class For SEGMENT call, address of SEGMENT class For OTHER call, address of OTHER class
2	Calling routine number as follows: 12 (X'0C') DFSDLA30 for DFSCTSE0 only, CHNG call 32 (X'20') DFSDLA30 for DFSCTSE0 only, AUTH call
3	Return code from prior routines
4	For details of the format of this storage area, see the prolog in the sample routine (IMS.TMSOURCE; member name is DFSCTSE0).
7	Address of source CTB or zeros. **Recommendation:**The contents of this register vary depending on the type of call to the exit routine and on the environment from which the call was made. It is recommended that applications not require the contents of this register for processing.
9	Address of PST.
10	Address of transaction code or resource name.
11	Address of SCD.
13	Address of save area. The exit routine must not change the first three words.
15	Entry point of exit routine.

Contents of Registers on Exit

Upon return, all registers must be restored except for register 15, which must contain one of the following return codes to indicate the success or failure of the user's authorization to issue a CHNG call.

Return Code	Meaning
0	IMS accepts the CHNG call.
4	The resource is not protected.
8	The user is not authorized.
Positive	IMS rejects the CHNG call.
Negative	User is authorized. The negative value is the complemented address that points to user data provided by RACF (AUTH call).

Chapter 56. Shared Printer Exit Routine (DFSSIML0)

This chapter describes the Shared Printer exit routine.

About This Routine

To acquire SLU 1, or 328X BSC/VTAM printers that are defined to IMS as shared, the IMS message router activates a Shared Printer exit routine. This is a routine that you write to decide whether a terminal that is unavailable can be automatically acquired by IMS or an AOI application program. The Shared Printer exit routine should return the name of the AOI application program.

A Shared Printer exit routine is not necessary to use shared printing. If no exit routine exists, the message router simulates a /OPN command when the terminal is defined as shared.

Attributes of the Routine

Table 104 shows the attributes of the Shared Printer exit routine.

Table 104. Shared Printer Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSSIML0.
Link editing	Follow the guidelines described in “Link-Editing the Routines” on page 5.
Including the routine	No special steps are required to include this routine.
IMS callable services	To use IMS Callable Services with this routine, you must issue an initialization call (DFSCSII0) to obtain the Callable Service token and a parameter list in which to build the function-specific parameter list for the desired callable service. Use the ECB found in register 9 for the DFSCSII0 call. **Related Reading:**For details on using the initialization call, see “Step 2. Initializing IMS Callable Services (DFSCSII0)” on page 11. This exit is automatically linked to DFSCSI00 by IMS. No additional linking is required to use IMS Callable Services.
Sample routine location	IMS.TMSOURCE (member name DFSSIML0).

Special Considerations

If you decide to write a Shared Printer exit routine, here are some things you need to know:

- If the exit routine returns a bad return code, it is disabled and message DFS2084 is sent to the master terminal operator. A bad return code, in this case, is a return code of 8 when no transaction name is in the area pointed to by register 1 or when the transaction name returned is invalid. After the exit routine has been disabled, a return code of 0 is assumed. For the exit routine to be enabled, IMS must be restarted.
- The exit routine must not issue any waits, OS macros, or SVCs.
- The exit routine may examine output destination but cannot modify it.
- The exit routine should return the name of the AOI application program in the field provided by the message router.

- The exit routine receives control of the messages after they are queued.
- Because the exit routine runs in the IMS control region, your installation must maintain security. Installation procedures should not let an unauthorized routine be linked into the nucleus.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the routine.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	**Contents**
1	Address of the area where the AOI transaction name is to be returned.
6	Address of CNT.
7	Address of CTB.
9	Address of CLB.
11	Address of SCD.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of exit routine.

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers except for register 15, which must contain one of the following return codes:

Return Code	**Meaning**
0	A SIMLOGON with the Q and RELRQ options is issued. This tells the other subsystem or VTAM application connected to the printer that IMS needs the printer.
4	No special processing is required. Normal processing continues.
8	The AOI transaction is activated. This transaction cannot be a conversational, Fast Path, remote, or password-protected transaction.

Chapter 57. Signoff Exit Routine (DFSSGFX0)

This chapter describes the Signoff exit routine.

You can write the Signoff exit routine to perform processing that complements the Signon exit routine (DFSSGNX0). You can also use this exit routine to reset the significant status for terminals during user signoff.

About This Routine

All attempts to sign off from ACF/VTAM terminals cause IMS to call this exit routine. The Signoff exit routine is also called if either RACF or the Signon/Signoff Security exit routine (DFSCSGN0) fails a signon attempt.

Recommendation: Although the Sign-On exit routine and this exit routine are optional, if you include one, you should also include the other to perform any cleanup operations that are necessary.

Table 105 shows the attributes of the Signoff exit routine.

Table 105. Signoff Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSSGFX0.
Link editing	Follow the guidelines described in "Link-Editing the Routines" on page 5.
Including the routine	If you want IMS to call the Signoff exit routine, include it in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.RESLIB.
IMS callable services	To use callable services with this routine, you must do the following: • Issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. For details on using the initialization call, see "Step 2. Initializing IMS Callable Services (DFSCSII0)" on page 11. • Use the current address ECB found at offset 0 for the DFSCSII0 call. • Link DFSCSI00 with your user exit.
Sample routine location	IMS.TMSOURCE (member name DFSSGFX0).

Restrictions

The Signoff exit routine cannot be used by LU 6.2 terminals.

Extended Recovery Facility (XRF) Considerations

Each time IMS calls the Signoff exit routine, the exit routine receives information on the XRF status of IMS. The exit routine can check this information and return the appropriate error message if necessary. IMS calls the exit routine if XRF tracking fails.

Resetting the Significant Status

You can use this exit to reset the significant status for a terminal in one of the following states:

Conversational
Exclusive
Test
Preset
MFS test
Response

A parameter passed to the exit routine indicates the status of the terminal or ETO user at sign off. You can reset the status in the output parameters.

Resetting Status When VTAM Manages Generic Resource Affinities

When you specify GRAFFIN=VTAM on the DFSDCxxx PROCLIB member, IMS automatically insures that all non-ISC terminal status conditions are reset for sessions using the IMS generic resource name before the Signoff or Logoff exit routine or before the next logon or signon to a failed and restarted IMS.

For **conversation mode**, IMS performs the equivalent of an `/EXIT` command for the conversation.

Related Reading: Refer to the *IMS/ESA Operator's Reference* for information about the `/EXIT command`.

For **full-function response mode**, IMS sends the response mode reply asynchronously.

For **Fast Path response mode**, IMS resets the response mode and discards the output reply message when a transaction is:

- Scheduled in global mode
- Scheduled in local mode, and the output is available when IMS attempts to reset status

Note that, regardless of whether the Fast Path output reply message is available, the log record indicates that the dequeue (reset of Fast Path response mode) is a result of the GRAFFIN=VTAM option. The log record uses the new flag shown below:

```
FLDQFLG DS X                  FLAG BYTE
FLDQFFDQ0 EQU X'10'           DEQ VIA GRAFFIN OR USER EXIT
```

Session termination during local mode Fast Path input scheduling or processing does not prevent VTAM from releasing affinity. However, IMS cannot terminate the Fast Path process. Therefore, it does not reset response mode before the next session is initiated. Input attempted in this case causes a DFS2162 to occur. The behavior is the same as when you specify GRAFFIN=IMS.

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the exit routine.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	Contents
R1	Address of Standard Exit Parameter List (Table 106)
R13	Save area address
R14	Return address to IMS
R15	Entry point address of exit routine

Table 106. Standard Exit Parameter List (Version 1)

Offset	Length	Description
+0	4	Address of standard exit parameter list version number
+4	4	Reserved
+8	4	Reserved
+12	4	Address of Signoff Exit parameter list

Table 107 lists the signoff parameters.

Table 107. Signoff Exit Parameter List

Offset	Length	Description
+0	4	Current ECB address.
+4	4	SCD address.
+8	4	Address of the user table created by initialization user exit DFSINTX0 or zero, if none.
+12	4	Address of USERID associated with Signoff.
+16	4	CLB address.
+20	4	Address of the STATUS_IN and STATUS_OUT vectors. The status vectors are mapped by the DFSSTCHK macro. For the contents of the STATUS_IN, vector see “Contents of STATUS_IN”. For the contents of the STATUS_OUT vector, see “Contents of STATUS_OUT” on page 390.

Contents of STATUS_IN

The input status vector is a 2-byte field that indicates the terminal’s significant status when the exit routine is called. The second byte of the field is reserved. The first byte of the field contains a value that indicates the significant status as follows:

Value	Description
X'80'	Conversation
X'40'	Exclusive
X'20'	Test
X'10'	Preset
X'08'	MSF Test
X'04'	Response

Contents of STATUS_OUT

The output status vector is a two-byte field that indicates changes to the significant status made by the exit routine. IMS uses the contents of STATUS_OUT as an indicator to exit a conversation and reset significant status. The default for this field is zeros, indicating that no significant status is reset.

The second byte of the field is reserved. The first byte of the field contains a value that indicates the significant status to be reset as follows:

Value	Description
X'80'	Exit conversation
X'40'	Reset exclusive
X'20'	Reset test
X'10'	Reset preset
X'08'	Reset MSF test
X'04'	Reset response

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers except for registers 15, which contains the return code

Register	Contents
15	One of the following return codes:

Return Code	Meaning
0	Normal return.
Negative value	The specified user message is sent to the terminal signing off. This message can be used to trigger an AOI facility following a signoff operation.

Chapter 58. Sign-On Exit Routine (DFSSGNX0)

This chapter describes the Sign-On exit routine. IMS calls the Sign-On exit routine for sign-on processing if ETO=Y is specified as an execution parameter. All attempts to sign-on to ACF/VTAM terminals if the Extended Terminal Option (ETO) feature is active cause IMS to call this exit routine. The Sign-On exit routine cannot be used by LU 6.2 terminals.

IMS calls the Sign-On exit routine before RACF validation (if requested) is performed and before the Sign On/Off Security exit routine (DFSCSGN0) is called. This exit routine contains logic and function that complement the Sign On/Off Security exit routine. Review your use of the Sign On/Off Security exit routine to determine if the function that it provides is necessary or conflicts with the Sign-On exit routine.

Related Reading:

- For more information on ETO and LU 6.2, see *IMS/ESA Administration Guide: Transaction Manager*.
- For more information on the Sign On/Off Security exit routine, see “Chapter 59. Sign On/Off Security Exit Routine (DFSCSGN0)” on page 399.

About This Routine

You can write the Sign-On exit routine to:

- Select the user descriptor (based on the user ID, node name, or DFSUSER) that you want IMS to reference when building the control block structure for the user that is signing on. (For more information, see “Selecting a User Descriptor” on page 394.)
- Provide queue data for the user that is signing on. This could be data to override the queue data derived from the user descriptor. If the user descriptor is DFSUSER, the exit routine can also supply queue data to add additional LTERMs to the structure. (For more information, see “Providing Queue (LTERM) Data” on page 395.)
- Supply parameters that you want IMS to reference when building associated printer control block structures. (For more information, see “Supporting Associated Printing” on page 393.)
- Allow or disallow a sign-on attempt based on a maximum number of users, or according to any criteria that you specify.
- Specify, or override, autologoff parameter and autosignoff (ASOT) value.

For the latest version of DFSSGNX0, see the IMS.TMSOURCE library; member name is DFSSGNX0. If you write your own Sign-On exit routine or modify the sample, you must include the portion of the sample exit routine (or the equivalent logic) that removes extraneous blank fields that RACF (if used) creates. (When the Sign-On exit routine is **not** included in the system, internal IMS logic removes these extraneous blank fields.) The sample exit routine also provides an example of associated printing.

When the Sign-On exit routine (DFSSGNX0) is not included in the system and the MFS formats for the DFS3649 message have not been modified, internal IMS logic removes these extraneous blank fields. If the MFS formats for the DFS3649 message have been modified, corresponding changes to the logic in the Sign-On exit routine that removes the extraneous blank fields may be necessary. This logic is included in the Sign-On exit routine so that adjustments can be made when changes are made to the DFS3649 MFS formats.

The Sign-On exit routine and the Output Creation exit routine (DFSINSX0) are corequisite exit routines, under the following conditions. If you provide one exit routine to supply queue data for additional LTERMs, you must provide the other exit routine also. They both create the user control block structure and related LTERMs (including multiple LTERMs for a user): the Sign-On exit routine using the user ID and the Output Creation exit routine using an LTERM name. Both exit routines must have the same logic so that the structure created is identical, regardless of which exit routine created it.

You can use the Sign-Off exit routine (DFSSGFX0) to complement any processing that the Sign-On exit routine performs.

User ID

The Sign-On exit routine informs the external subsystem of the user ID associated with the transaction input message. The user ID can be one of the following:

- The inputting LTERM name if the terminal user is not signed on
- The ID of the terminal user
- The RACF/user-authorized user ID associated with a non-message driven BMP or CPIC application
- The PSB name specified on the job card

IMS determines the user ID in the following order.

For CPIC application:

1. RACF ID if the ACEE is cloned in the dependent region
2. PSTBUSER if the field does not contain binary zeros or blanks
3. PSTUSID if the field does not contain blanks
4. PSTSYMB0 if the field does not contain blanks
5. PDIRSYM

For a message driven BMP that has done a GU, or IFP that has done a GU, or MPP:

1. PSTUSID if the field does not contain blanks
2. PSTSYMB0 if the field does not contain blanks
3. PSTBUSER if the field does not contain binary zeros or blanks
4. PDIRSYM

For message driven BMP that has not done a GU or IFP that has not done a GU:

1. PSTBUSER if the field does not contain binary zeros or blanks
2. PDIRSYM

For non-message driven BMP:

1. PSTBUSER if the field does not contain binary zeros or blanks and the DFSDCxxx proclib member specifies BMPUSID=USERID
2. PDIRSYM

When a dependent region connection is initially established, the Sign-On exit routine is activated before a thread is created by the Create Thread exit routine. All subsequent Sign-On requests result in the exit routine being activated after a thread is created. For example, Sign-On is activated for each message processed during a single scheduling, whether or not the messages are separated by commit processing.

Extended Recovery Facility (XRF) Considerations

IMS calls the Sign-On exit routine in the XRF alternate system for a type 1 session with ETO. When IMS calls the exit routine in the alternate system, the exit routine is not allowed to change **anything** related to the terminal or user structures, including fields that the exit routine can normally change.

Each time IMS calls the Sign-On exit routine, the exit routine receives information on the XRF status of IMS.

Attributes of the Routine

Naming the Routine

You must name the Sign-On exit routine DFSSGNX0.

Assembling and Loading the Routine

A sample Sign-On exit routine is provided in IMS.TMSOURCE. Alternatively, you can write your own exit routine. You can assemble the sample exit routine or one that you write (using the standard IMS macro and copy files), and include it in an authorized library in the JOBLIB, STEPLIB, or LINKLIST library concatenated in front of the IMS.RESLIB. If the Sign-On exit routine is included, IMS automatically loads it each time IMS is initialized if ETO=Y (after the Initialization exit routine, DFSINTX0, has changed the ETO= parameter).

If you want associated printing, be sure to specify the following when you assemble the sample exit routine:

```
&ASSOCPRT SETC 'YES'
```

This specification ensures that the associated printing sample code is generated.

Supporting Associated Printing

Associated printing is the ability to direct application printer output to a printer logical unit (LU) name. This LU name is provided at either logon or sign-on time. If the Logon exit routine (DFSLGNX0) is written to detect LU names entered as logon user data, IMS passes these LU names to the Sign-On exit routine.

If you modify the DFS3649A MFS format to allow LU names to be entered as /SIGN ON user data, the Sign-On exit routine must be able to detect the LU names. If the user can enter LU names directly at sign-on, the exit routine must determine the queue name that is allocated to each printer. There should be a unique relationship between the user ID and the queue name. The Sign-On exit routine passes the queue name to IMS, which creates the control block structure. An application program can use the same algorithm to determine the queue name when the application processes a transaction scheduled for a particular user ID.

The exit routine must insert a period (.) in the sign-on user verification string (UVS) after building the associated printer buffer.

Selecting a User Descriptor

If the user control block structure already exists for the user that is signing on, IMS searches for the user structure and passes the addresses of the existing nodename structure along with the address of any existing user ID structure to the exit routine in the parameter list ESPQBTAB.

The exit routine can determine whether to use the user ID structure or the nodename structure by examining the passed structure without making an explicit IMS Callable Service routine call to find the nodename user structure.

If the exit routine chooses the nodename as the user structure name, the user ID is hashed to a non-SPQB user hash table.

If no user control block structure exists, you can select a user descriptor by using the USERD= keyword, write the Sign-On exit routine to select the user descriptor, or let IMS select a descriptor. Figure 21 shows the search order that IMS uses to select the user descriptor.

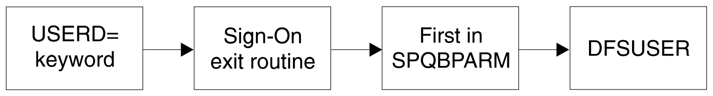


Figure 21. User Descriptor Selection Order

You can use the USERD= keyword (entering it as user data with the `/SIGN ON` command) to select the user descriptor. If your Sign-On exit routine does not choose a user descriptor, IMS uses the user descriptor requested via the USERD= parameter.

The Sign-On exit routine is called with the parameter list SPQBPTRS, which contains the address of the USERD= keyword specified, and the addresses of the user ID descriptor, node name descriptor, and DFSUSER descriptor. The exit routine can choose among these descriptors by specifying the descriptor's address in the USEQUSED field of the USEQDATA DSECT.If the exit routine selects one of these user descriptors, IMS uses it to create the user control block structure. (A user descriptor that the exit routine specifies overrides any descriptor specified on the USERD= keyword.)

The exit routine can also create an arbitrary user structure name by specifying the name in the eight-byte USEQUSTN field in the USEQDATA parameter list. IMS creates the user structure with the name from this field and stores the returned name in the SPQB user hash table. Security is based on the original userid that the user signed on with and is stored in the non-SPQB user hash table.

If no user descriptor is specified on the USERD= keyword and the Sign-On exit routine does not return the address of a user descriptor, IMS selects the first descriptor address that it finds in the SPQBPTRS among the user ID descriptor, node name descriptor, and the DFSUSER descriptor, respectively. IMS uses this descriptor to create the user control block structure.

If none of these methods returns a user descriptor, IMS uses DFSUSER to create the user structure. If no user descriptor can be found, including DFSUSER, IMS rejects the sign-on request.

Regardless of how the user descriptor is chosen, only DFSUSER or descriptors associated with the user ID or node name are valid. There is no user-based output security if the selected descriptor is the node name descriptor.

Providing Queue (LTERM) Data

Depending on the user descriptor selected, the exit routine can provide queue data. If the exit routine returns data that it was not authorized to return, IMS rejects the sign-on request.

Figure 22 identifies four cases which describe what data the Sign-On exit routine can supply. The four cases are based on whether the user structure exists and whether DFSUSER or a non-DFSUSER descriptor is selected. For this exit routine, non-DFSUSER descriptors are descriptors based on the user ID or node name. Each case is discussed in the sections that follow. (If the exit routine does not exist, refer to “Selecting a User Descriptor” on page 394.)

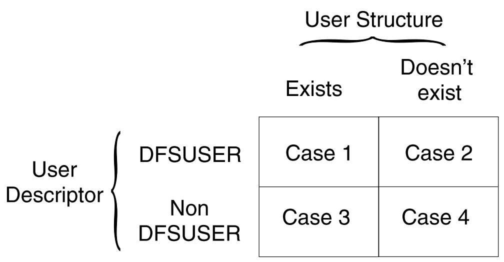


Figure 22. Case Numbers Identifying What Data DFSSGNX0 Can Provide

Case 1

The Sign-On exit routine is called using the descriptor, DFSUSER, that was used to create the user control block structure. The exit routine can:

- Supply queue data (except LTERM names) to override data of the existing structure
- Provide data for additional LTERMs, if it supplies the data for the existing LTERMs first and in the order in which they are chained

IMS verifies the additional LTERMs that are specified (but are not in the existing user structure) against the LTERMs that already exist in the system. If an LTERM that is specified as an additional LTERM already exists in the system, IMS assumes that this LTERM has been assigned to a different user, and it is not made part of the user structure of the user that is signing on. If this is the only LTERM that the descriptor or the Sign-On exit routine specifies for this user, IMS rejects the sign-on attempt.

Case 2

If DFSUSER is selected and no user control block structure exists, the Sign-On exit routine:

- Can supply any queue data desired (including LTERM names)

If the exit routine does not provide queue data, one LTERM (named for the user ID) is created. If **any** queue data is passed, this default user ID LTERM is not created and must be specified in the queue data if it is desired.

IMS verifies the additional LTERMs that are specified against the LTERMs that already exist in the system. If an LTERM that is specified already exists in the system, IMS assumes that this LTERM has been assigned to a different user, and it is not made part of the user structure of the user that is signing on. If this is the only LTERM that the descriptor or exit routine specifies for this user, IMS rejects the sign-on attempt.

Case 3

The Sign-On exit routine is called with the same non-DFSUSER descriptor that was used to create the user control block structure (either the user ID or node name descriptor). The exit routine:

- Can supply any queue data (except LTERM names) to override data of the existing structure
- Cannot provide data for additional LTERMs

IMS verifies the LTERMs that are specified in the descriptor (but are not in the existing structure) against the LTERMs that already exist in the system. If an LTERM is specified in the descriptor but is not in the existing structure, IMS assumes that this LTERM has been assigned to a different user and deleted. The LTERM is given back to the user and is made part of the user structure of the user that is signing on.

Case 4

If a non-DFSUSER descriptor is selected and no user control block structure exists, the Sign-On exit routine:

- Can supply queue data (except LTERM names) to override data that the descriptor provides
- Cannot provide data for additional LTERMs

IMS verifies the LTERMs specified in the descriptor against the LTERMs that already exist in the system. If an LTERM that is specified in the descriptor already exists in the system, IMS assumes that this LTERM has been assigned to a different user, and it is not made part of the user structure of the user that is signing on. If this is the only LTERM that the descriptor or exit routine specifies for this user, IMS rejects the sign-on attempt.

Using IMS Callable Services with This Routine

To use IMS Callable Services with this routine, you must do the following:

- Issue an initialization call (DFSCSII0) to obtain the Callable Service token and a parameter list in which to build the function-specific parameter list for the desired callable service. For details on using the initialization call, see “Step 2. Initializing IMS Callable Services (DFSCSII0)” on page 11.
- Use the current address ECB found at offset 0 for the DFSCSII0 call.
- Link DFSCSI00 with your user exit.

Communicating with IMS

IMS uses the entry and exit registers, as well as parameter lists, to communicate with the exit routine.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

MRegister	**Contents**
R1	Address of Standard Exit Parameter List (Table 108)
R13	Save area address
R14	Return address to IMS
R15	Entry point address of exit routine

Table 108. Standard Exit Parameter List (Version 1)

Offset	Length	Description
+0	4	Address of standard exit parameter list version number
+4	4	Reserved
+8	4	Reserved
+12	4	Address of Signon Exit parameter list

Table 109 lists the signon exit parameters.

Table 109. Signon Exit Parameter List

Offset	Length	Description
+0	4	Current ECB address.
+4	4	SCD address.
+8	4	Address of the user table created by Initialization User Exit routine DFSINTX0 or zero, if none.
+12	4	Address of a buffer for use by your user exit to return user descriptor and queue data exit parameters. For additional details on the content and the format, refer to the prolog in the sample routine. Zero for a static terminal. Zero if processing on an XRF alternate system. Zero if processing `/SIGN ON ETO STSN` device. The USEQDATA DSECT is provided for parameter area mapping.
+16	4	Address of a buffer for use by your user exit to return associated printer exit parameters. For additional details on the content and the format, refer to the prolog in the sample routine. Zero, if processing on an XRF alternate system.
+20	4	Address of a parameter list created by one of the following: • Session initiation from LOGON data. • Input from the `/SIGN ON` command. For additional details on the content and the format, refer to the prolog in the sample routine. Zero, if processing XRF alternate.

Table 109. Signon Exit Parameter List (continued)

Offset	Length	Description
+24	4	Address of a parameter list, which contains pointers to available user control block structures (SPQBs) and default autosignoff values. For additional details on the content and the format, refer to the prolog in the sample routine. Zero for a static terminal. Zero if processing on an XRF alternate system. Zero if processing /SIGN ON ETO STSN device.
+28	4	CLB address.
+32	4	Table of existing user structures. For additional details on the content and the format, refer to the prolog in the sample routine.
+36	1	Flag byte. For a description of the content, refer to the prolog in the sample routine.
+37	1	Reserved.
+38	1	Reserved.
+39	1	Reserved.

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers except for registers 15, which contains the return code.

The content of register 15 will be ignored if processing is on an XRF alternate system.

Register	Contents
15	One of the following return codes:

Return Code	Meaning
0	IMS continues SIGNON processing.
4	IMS rejects the SIGNON attempt. The SIGNON required message, DFS3649, is resent to the terminal with some added information indicating the reason for rejection.
Negative	The same as return code 4, but IMS sends the specified user message instead of DFS3649.

Using the Sample Sign-on Exit Routine (DFSSGNX0)

For the latest version of DFSSGNX0, see the IMS.TMSOURCE library; member name is DFSSGNX0.

Chapter 59. Sign On/Off Security Exit Routine (DFSCSGN0)

This chapter describes the Sign On/Off Security exit routine. You can use this exit routine to verify a user's ID and password.

Related Reading:

- This exit routine can conflict with the Sign-On exit routine (DFSSGNX0). See "Chapter 58. Sign-On Exit Routine (DFSSGNX0)" on page 391 for information on DFSSGNX0.
- The Transaction Authorization and the Security Reverification exit routines provide additional user verification for transactions. See "Chapter 62. Transaction Authorization Exit Routine (DFSCTRN0)" on page 411 and "Chapter 55. Security Reverification Exit Routine (DFSCTSE0)" on page 383 for more information.

About This Routine

You can use the Sign On/Off Security exit routine with or without RACF to verify the user ID and password. If both the RACF option and the Sign On/Off Security exit routine are selected in the IMS system definition, IMS calls this exit routine after RACF /SIGN ON verification has been performed. If the `/SIGN ON` request is rejected by RACF, IMS does not call this exit routine. If the RACF option is not selected in the IMS system definition, you can use this exit routine to verify the user's identification and passwords at `/SIGN ON` time.

If ETO=Y is specified as an execution parameter, the Sign-On exit routine (DFSSGNX0) performs sign-on processing before IMS calls RACF or the Sign On/Off Security exit routine. If the Sign-On exit routine rejects the signon attempt, IMS does not call the Sign On/Off Security exit routine.

Related Reading: For more information on the Extended Terminal Option (ETO) feature, see *IMS/ESA Administration Guide: Transaction Manager*.

If shared queues are active and the security environment for a transaction is created on the back-end IMS subsystem, IMS does not call this exit routine.

The Sign On/Off Security exit routine should have access to a table of valid user IDs and the passwords associated with each ID. The exit routine should note successful `/SIGN ON`s to prevent additional attempts to perform the `/SIGN ON` function. When the `/SIGN OFF` command is executed, the exit routine should mark that user ID as available for `/SIGN ON`. For logging purposes, the exit routine can also place information into the data portion of the user verification string that is passed to the exit.

If you plan to use the Sign-On exit routine, review how you use the Sign On/Off Security exit routine to determine if the function that this exit routine provides is necessary or might even conflict with the Sign-On exit routine.

Table 110 shows the attributes of the Sign On/Off Security exit routine.

Table 110. Sign On/Off Security Exit Routine Attributes

Attribute	Description
IMS environments	TM environments
Naming convention	You must name this exit routine DFSCSGN0.

Table 110. Sign On/Off Security Exit Routine Attributes (continued)

Attribute	Description
Link editing	Follow the guidelines described in “Link-Editing the Routines” on page 5.
Including the routine	No special steps are required to include this routine.
IMS callable services	To use callable services with this routine, you must issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. For details on using the initialization call, see “Step 2. Initializing IMS Callable Services (DFSCSII0)” on page 11. Use the ECB found in register 9 for callable services. This exit is automatically linked to DFSCSI00 by IMS. No additional linking is required to use callable services.
Sample routine location	IMS.TMSOURCE (member name DFSCSGN0).

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of Registers on Entry

Upon entry to the exit routine, all registers must be saved using the save area provided. The registers contain the following:

Register	Contents
0	`/SIGN` function (ON or OFF): 0 `/SIGN ON` 1 `/SIGN OFF` 2 `/SIGN ON` in XRF alternate system. 3 `/SIGN OFF` in XRF alternate system.
1	Pointer to the variable-length user verification string, if the SIGN function is `/SIGN ON`. The string format is LLZZ (4 bytes), followed by the text, starting with the first character of the user ID. Address of the user ID if the SIGN function is `/SIGN ON`in an XRF environment. Insignificant if the SIGN function is `/SIGN OFF`.
7	Address of source CTB or zeros. **Recommendation:**The contents of this register vary depending on the type of call to the exit routine and on the environment from which the call was made. It is recommended that applications not require the contents of this register for processing.
9	Address of ECB.
11	Address of SCD.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of exit routine.

Contents of Registers on Exit

Upon return to IMS, all registers must be restored except for register 15, which contains one of the following return codes:

Return Code	Meaning
0	IMS accepts the `/SIGN ON`

Return Code	Meaning
Positive	IMS rejects the `/SIGN ON`. IMS sends message DFS2467 if sign-on is not required and message DFS3649 if sign-on is required. This return causes a “BY IMS EXIT” to be appended to the message to indicate that the exit routine caused the return code.
Negative	IMS rejects the `/SIGN ON` command and sends a user-defined message. The message number is complemented into a message number. This number must be less than -24, otherwise a DFS2467 message is sent instead. You must list the absolute value of this message number in the User Message Table, DFSCMTU0. (See “Chapter 28. User Message Table (DFSCMTU0)” on page 213 for more information on the user message table.)

Exception: The exit routine does not check this return code on return from RACF or during `/SIGN OFF` processing.

Chapter 60. Terminal Routing Exit Routine (DFSCMTR0)

This chapter describes the Terminal Routing exit routine. Message routing between IMS systems is based on logical terminals (LTERMs) and transactions, just as it is in a single-IMS system. In a multiple-system configuration, LTERMs and transactions can be a part of any of the systems in the configuration.

Routing of messages is automatic, because each system in the configuration knows, via system definition, which transactions and logical terminals are local to it, and which ones it can reference by message switches, program-to-program switches, or transactions.

In an IMS system, additional control of message routing can be accomplished with the Routing exit routine described here. This is true for both MSC and non-MSC IMS systems. This exit routine does not apply to ISC.

Recommendation: Whenever possible, you should use the Input Message Routing exit routine (DFSNPRT0) instead of the Terminal Routing exit routine. If both routines are included, IMS calls DFNSPRT0. For more information on DFSNPRT0, see “Chapter 38. Input Message Routing Exit Routine (DFSNPRT0)” on page 297.

About This Routine

The Terminal Routing exit routine can change the destination of an input message to any local or remote destination. It receives control from IMS when the first segment of an input message is received from a terminal and before the analyzer determines the destination of the message. Refer to the data and register example.

Register 1 points to the edited (MFS or Basic Edit) segment, and register 2 points to another 8-byte area that contains a destination name. Register 3 contains the length of the destination name pointed to by register 2. Because registers 1 and 2 are not equal, this routine can affect routing in several different ways. With any of these methods, data within the segment pointed to by register 1 can be changed, but the length of the segment cannot be changed.

Related Reading: For more information on MFS edit options, see the section on input message formatting options in *IMS/ESA Application Programming: Transaction Manager*.

The techniques for use with the Terminal Routing exit routine are:

- Change the destination name in the area pointed to by register 2. This causes the edited segment to be routed to the new destination with no change in the original transaction code or data. The Terminal Routing exit sample routine (DFSCMTR0 in IMS.TMSOURCE) uses this technique.
- Change the name in the segment pointed to by register 1. This does not cause rerouting but gives the application program a different transaction code.
- Change register 2 to point to an area containing a destination name. This results in an operation similar to the first technique described.
- Change both register 2 and the data pointed to by register 1. This method must be used when the transaction code in the segment is to be the same as the destination.

If the input from the terminal is a mixture of upper case and lower case, the destination field pointed to by register 2 is translated and presented to the exit routine in upper case only. The segment itself, pointed to by register 1, is presented to the exit routine in its original form.

This exit routine, in combination with the Link Receive Routing exit routine, can be used to reduce the number of remote transactions that must be defined.

Related Reading: For more information about the Link Receive Routing exit routine, see "Chapter 46. MSC Link Receive Routing Exit Routines (DFSCMLR0/DFSCMLR1)" on page 341

For example, if there are 40 different transactions that execute in a specific remote system and all transactions have the same attributes (nonconversational, noninquiry, response mode, etc.), the exit routine can:

1. Define one remote transaction that represents all 40, such as XX.
2. Change the destination name pointed to by register 2 to the name that represents all 40 transactions in the remote system, when a message for any one of the 40 remote transactions is detected.

Table 111 shows the attributes of the Terminal Routing exit routine.

Table 111. Terminal Routing Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSCMTR0.
Link editing	Follow the guidelines described in "Link-Editing the Routines" on page 5. If this exit routine uses the IWAIT macro or calls other exit routines that can use the IWAIT macro, the exit routine must be reentrant.
Including the routine	Include this routine as a stand-alone module in IMS.RESLIB or IMS.STEPLIB.
IMS callable services	To use IMS callable services, you must issue an initialization call (DFSCSI00) to obtain the callable services token and a parameter list in which to build the function-specific parameter list for the desired callable service. Use the ECB found in register 9 for the DFSCSII0 call. For details on using the initialization call, see "Step 2. Initializing IMS Callable Services (DFSCSII0)" on page 11.
Sample routine location	IMS.TMSOURCE (member name DFSCMTR0).

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of Registers on Entry

Register	Contents
1	Address of the segment.
2	Address of an eight-byte name field. The exit must change this field or change this pointer to another eight-byte name field if another destination is desired.
3	Length of destination pointed to by register 2. Register 3 must reflect the length of the destination pointed to by register 2 when exiting DFSMTR0.
7	Address of the CTB for the terminal from which the message was read.
8	Address of the CTT for the terminal from which the message was read.

Register	Contents
9	Address of the CLB for the terminal from which the message was read.
11	Address of the SCD.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of exit routine.

Contents of Registers on Exit

Before returning to IMS, the exit routine must restore all registers except for register 15, which must contain the following:

Register	Contents
15	0

Chapter 61. Time-Controlled Operations (TCO) Exit Routine (DFSTXIT0)

The TCO exit routine inserts messages in the message queue at a specific time for processing.

About This Routine

The TCO exit routine inserts messages that are the commands, transactions, and message switches that you specify in the time schedule requests and message sets that make up a script member. The TCO exit routine passes any data found in columns 56 through 71 of the time schedule request to IMS to be processed.

You do not have to write your own exit routine. You can schedule predefined commands, transactions, and message switches at predefined times with DFSTXIT0, the TCO exit routine IMS supplies. If you do write your own, you can write it in COBOL or assembler.

Restriction: PL/I and C language exit routines are not supported. Cobol routines running under Language Environment for OS/390 are not supported.

This routine cannot be used in a DBCTL environment.

Table 112 shows the attributes of the Time-Controlled Operations (TCO) exit routine.

Table 112. Time-Controlled Operations (TCO) Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	If you are writing your own routine, you can name it anything you wish. If you are using the IMS-supplied routine, use the name DFSTXIT0.

Link editing	You should write, compile, and link-edit the routine as **serially reusable** (REUS).
	
	Here is an example of a routine named MYEXIT being link-edited to DFSTDLI0.
	
	`//XIT       JOB 1, MSGLEVEL=1`
	`//LINK      EXEX PGM=IEWL, PARM=REUS`
	`//SYSUT1    DD UNIT=SYSDA,SPACE=(TRK,(20,20))`
	`//SYSPRINT  DD SYSOUT=A`
	`//SYSLMOD   DD DSN=IMS.RESLIB,DISP=SHR`
	`//INLIB     DD DSN=IMS.OBJ,DISP=SHR`
	`//SYSLIN    DD *`
	`    INCLUDE  INLIB(MYEXIT)`
	`    INCLUDE  INLIB(DFSTDLI0)`
	`    NAME     MYEXIT(R)`
	`/*`
	
	In the example above, IMS.RESLIB is an authorized library that contains all load modules. IMS.OBJ is a library that contains all object modules. The above JCL expects to find the object modules of the exit routine (MYEXIT) and the TCO Language Interface module (DFSTDLI0) in IMS.OBJ and places the result of the link-edit into IMS.RESLIB.
	
	After you've compiled and tested your routine (or if you are using the routine supplied by IMS), you must link-edit the exit routine with the TCO Language Interface module (DFSTDLI0) and place them into IMS.RESLIB.

Table 112. Time-Controlled Operations (TCO) Exit Routine Attributes (continued)

Attribute	Description
Including the routine	To load and execute the routine, it must be referred to in a time schedule request in the script member that is executing. **Related Reading:** For more information about time schedule requests and script members, see *IMS/ESA Operations Guide*. The following is an example of a time schedule request in a script member that wants the routine “MYEXIT” to be executed. `*TIME 1200 MYEXIT` • Columns 1-5 contain the Identification field. '*TIME' is in this field. • Columns 7-10 contain the initial dispatch time. In this example it is 12:00 p.m. • Columns 12-19 contain the name of the exit routine, left-justified and padded with blanks. The name in this example is 'MYEXIT'.
IMS callable services	This exit is not eligible to use IMS Callable Services.
Sample routine location	IMS.TMSOURCE (member name DFSTXIT0).

Communicating with IMS

IMS uses the entry and exit registers, and parameters to communicate with the exit routine.

Contents of Registers on Entry

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Register	**Contents**
1	Address of a parameter list that contains the address of the program communication block (PCB) used in the exit routine calls.
10	Reserved for TCO.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of exit routine.

The PCB

The program communication block (PCB) contains the actual scheduling time for the time-initiated message processing. It is in the PCBTIME field (PCB + 16). Under most circumstances, this is the same time as the time initiated request. In very busy systems, however, that actual scheduled time may differ from the schedule request. For example, if you request your exit routine to be scheduled at 12:01 and a busy system prevents it from being scheduled until 12:03, the PCB contains 12:03.

DL/I Calls

The calls you can use in this exit routine are:

GU	Get the message that caused scheduling.
ISRT	Put a message segment into the queue for processing.
PURG	Terminate the prior segments as a message and insert the first segment on the next message (if an I/O area is provided).
GSCD	Get the address of the IMS system contents directory. The address is returned in the first word of the I/O area, which must begin on a word boundary.

The TCO exit routine calls the TCO Language Interface module (DFSTDLI0) to process the calls mentioned above. You can call DFSTDLI0 or CBLTDLI0 (for COBOL) to process the call.

You must pass a parameter list with the call in standard DL/I format (for example, register 1 contains the address of a 2- or 3-word parameter list whose end is indicated by a X'80' in the high-order byte). The PURG call can have two or three parameters. The other calls require three parameters.

The parameter list consists of:

1. The call function
2. The address of the I/O PCB
3. The address of the I/O area (optional with PURG)

Status Codes

A blank status code is returned to the exit routine after a successful call.

The following status codes can be returned to the exit routine after an unsuccessful call:

AB	The call didn't specify an I/O area.
AD	The function parameter on the call is invalid or is not supplied. The functions recognized by TCO are GU, ISRT, PURG, and GSCD.
AX	The I/O PCB name was invalid.
AZ	An ISRT or PURG call with an unacceptable message count was issued.
QC	There are no additional input messages to process for this time request.
QX	The ISRT or PURG call could not be processed because of insufficient storage.

Message Formats

A GU call always retrieves a message in one of these formats:

- A 20-byte example:

LL 20	ZZ	16-byte user data found in time required (col. 56-71 of time schedule request)
2	2	16

- An 8-byte example:

LL 8	ZZ	Address of the first segment of a message set (col. 56-59 of the time schedule request was ****)
2	2	4

- An example in which the address of a message set is retrieved:

Next Segment Pointer	LL	ZZ	Data
4	2	2	Variable Length

The last message of the message set contains binary zeros in the “next segment” field.

If the message set is broken into individual messages and segments (by the use of a space and an **S** in column 72), this is shown in the ZZ field of each segment. The values are:

Value	**Meaning**
0001	First segment of a message
0000	Middle segment of a message
0002	Last segment of a message
0003	First and last (only) segment of a message

Chapter 62. Transaction Authorization Exit Routine (DFSCTRN0)

This chapter describes the Transaction Authorization exit routine. The Transaction Authorization exit routine works with the Security Reverification exit routine (DFSCTSE0) and the Sign On/Off Security exit routine (DFSCSGN0) to check an individual user ID for authority to use a transaction.

Related Reading: See “Chapter 55. Security Reverification Exit Routine (DFSCTSE0)” on page 383 and “Chapter 59. Sign On/Off Security Exit Routine (DFSCSGN0)” on page 399 for more information on these exit routines and the transaction security they provide.

About This Routine

This exit routine can be used with or without RACF to verify that the user’s ID is authorized to run a transaction. If both the RACF option and the Transaction Authorization exit routine are selected in the IMS system definition, the exit is activated after RACF verifies the transaction. If the transaction request is rejected by RACF, the exit is not called. If the RACF option is not selected in the IMS system definition, this exit routine can be used to verify the user’s authorization and the password, if required, for that transaction.

The exit routine should have access to a table of valid user IDs, and the passwords and transactions associated with each valid user ID.

If you want to generate your own messages for the routine, you need to make the message number negative in register 15 to issue a specific message, and you need to list the absolute value of this message number in the User Message Table, DFSCMTU0. For details, see the description of the User Message Table presented in DFSCMTU0.

If you do not list this message in the User Message Table, message DFS060I is issued instead of the message you wanted to send.

Table 113 shows the attributes of the Transaction Authorization exit routine.

Table 113. Transaction Authorization Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSCTRN0.
Link editing	Follow the guidelines described in “Link-Editing the Routines” on page 5. The Security Reverification exit routine (DFSCTSE0) can be written as part of DFSCTRN0. See “Chapter 55. Security Reverification Exit Routine (DFSCTSE0)” on page 383 for more information about writing DFSCTSE0.
Including the routine	To implement this exit routine, specify TYPE=TRANEXIT in the SECURITY macro during Stage 1 input.

Table 113. Transaction Authorization Exit Routine Attributes (continued)

Attribute	Description
IMS callable services	To use callable services with this routine, you must issue an initialization call (DFSCSII0) to obtain the callable service token and a parameter list in which to build the function-specific parameter list for the desired callable service. Use the ECB in register 9 for the DFSCSII0 call. For details on using the initialization call, see “Step 2. Initializing IMS Callable Services (DFSCSII0)” on page 11. This exit is automatically linked to DFSCSI00 by IMS. No additional linking is required to use callable services.
Sample routine location	IMS.TMSOURCE (member name DFSCTRN0).

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of Registers on Entry

Upon entry to the exit routine, all registers must be saved using the save area provided. The registers contain the following:

Register	Contents
0	Register contents is dependent upon what is processed: To process a deferred program-to-program switch (R2 = 8), or DL/I CHNG call (R2 = C), then R0 = pointer to user ID (PSTUSID). To process receipt of a transaction received on an MSC link from a remote IMS system (R2 = 4), then R0 = pointer to user ID in the security prefix of the message. This exit routine is only called when R2 = 4 and the received message is sent using directed routing. For more information on directed routing, see “Communicating with Other IMS TM Systems Using MSC” in *IMS/ESA Application Programming: Transaction Manager*.
1	Address of the password or 0: For AUTH call, address of GENERIC class For TRAN call, address of TRAN class For FIELD call, address of FIELD class For DATABASE call, address of DATABASE class For SEGMENT call, address of SEGMENT class For OTHER call, address of OTHER class
2	Calling routine number, as follows:

Number	**Name**
0	Transaction input from terminal
4	Transaction from remote MSC system
8	Deferred conversation program-to-program switch
C	CHNG DL/I call
10	/SET command
14	/LOCK command
1C	/RELEASE command
20	AUTH call
24	LU 6.2 AUTH call
28	Transaction input from OTMA

Register	Contents
3	Address of storage area. For details of the format of this storage area, see the prolog in the sample routine (IMS.TMSOURCE; member name is DFSCTRN0).

Register	**Contents**
7	Address of source CTB or zeros. **Recommendation:**The contents of this register vary depending on the type of call to the exit routine and on the environment from which the call was made. It is recommended that applications not require the contents of this register for processing.
9	Address of ITASK control block as follows:

If Register 2=	**Then register 9= address of**
0	CLB
4	LLB
8	PST
C	PST
10	CLB
14	CLB
1C	CLB
20	PST
24	CLB

10	Address of transaction code or resource name.
11	Address of SCD.
13	Address of save area. The exit routine must not change the first three words.
14	Return address to IMS.
15	Entry point of exit routine.

Contents of Registers on Exit

Upon return to IMS, all registers must be restored except for register 15, which must contain one of the following return codes to indicate the success or failure of the user's authorization to a transaction.

Return Code	**Meaning**
0	Accept the transaction.
4	The resource is not protected.
8	The user is not authorized.
Positive	Reject the transaction and send DFS2469 message with register 15 halfword contents as a subcode if the transaction is entered from a terminal. The IMS system translates the subcode of message DFS2469 as follows:

Subcode	**Meaning**
08	Transaction not authorized (user is not authorized).
12	RACF is not active.
16	Invalid exit return code.
36	No password (password reverification is required, but no password was supplied).
40	Wrong password (password reverification failed).
Others	IMS exit CD (subcode generated by IMS exit).

Negative	For Resource Authorization: User is authorized. The negative value is the complimented address that points to user data provided by RACF (AUTH call).
Negative	For Transaction Authorization: Reject the transaction and send a user-defined message number, if appropriate, to the user. If the calling routine is DFSCON10 or DFSDLA30, no message is sent, but an A4 status code is passed to the application program. The message number passed must be less than -24.

Chapter 63. Transaction Code (Input) Edit Routine (DFSCSMB0)

This chapter describes the Transaction Code (Input) Edit routine.

About This Routine

During system definition, IMS gives you the ability to specify one or more Input edit routines in the IMS control program nucleus. This allows you to edit input messages before they are enqueued for scheduling. When IMS is executed, this edit function is performed in addition to IMS Basic Edit or MFS (Message Format Service) editing. The edit routine is entered before translating to upper case. You can specify at system definition up to 255 editing routines and you can also specify, by transaction type, which edit routine is to be used.

The edit routine must store the edited message segment to be returned to IMS in the buffer addressed by register 1. If the input was processed by the IMS Basic Edit, this buffer is always 10 bytes greater than the 2-byte binary count at the beginning of the message segment, and the message segment can be expanded or reduced to any desired size. The format of the edited message segment in the buffer upon return to IMS must be two bytes of binary count followed by bytes 3 and 4 unchanged from the original message and edited text.

If the input was processed by MFS, the length of this buffer is in the first two bytes of the buffer. No extra space is provided in this buffer for edit routines.

This edit routine is called only when a transaction is entered from a terminal; it is not called when the transaction is inserted via a program-to-program switch or for LU 6.2 terminals.

If specified, an Input edit routine gains control after each message data segment is processed by Basic Edit or MFS. Transaction code validity and security will have already been checked. If the transaction code is the only data in the message segment and the transaction is a conversational transaction, an edit routine is not entered.

Table 114 shows the attributes of the Transaction Code (Input) Edit exit routine.

Table 114. Transaction Code (Input) Edit Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSCSMB0.
Link editing	Follow the guidelines described in “Link-Editing the Routines” on page 5.
Including the routine	No special steps are required to include this routine.
IMS callable services	To use IMS Callable Services with this routine, you must do the following: • Issue an initialization call (DFSCSII0) to obtain the Callable Service token and a parameter list in which to build the function-specific parameter list for the desired callable service. For details on using the initialization call, see “Step 2. Initializing IMS Callable Services (DFSCSII0)” on page 11. • Use the ECB found in register 9 for the DFSCSII0 call. • Link DFSCSI00 with your user exit.

Table 114. Transaction Code (Input) Edit Exit Routine Attributes (continued)

Attribute	Description
Sample routine location	IMS.TMSOURCE (member name DFSCSMB0).

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of Registers on Entry

Upon entry to the edit routine, all registers must be saved using the save area provided. The registers contain the following:

Register	Contents
1	Address of the buffer location of the input message segment after translation to EBCDIC and after IMS Basic Editing. The first two bytes of the buffer contain the binary message length. The third byte of the buffer is binary zeros. The binary count includes the 4-byte prefix. If Basic Edit is used, the fourth byte of the message segment (Z2) is X'00'. If MFS is used, the fourth byte can contain either a X'01', X'02', or X'03' signifying that option 1, 2, or 3 respectively was selected for the message by the format designer. The fifth byte contains the first byte of the message text. If the input was processed by MFS, the length of this buffer is in the first two bytes of the buffer. No extra space is provided in this buffer for edit routines.
7	CTB address of the physical terminal from which the message is entered.
9	Address of CLB for the communication line from which the message is entered.
10	Address of SMB.
11	Address of SCD.
13	Address of save area. The first three words must not be changed.
14	Return address to IMS.
15	Entry point of edit routine. The entry point name and load module name for an edit routine must be the same as the name used for the edit routine in system definition.

Contents of Registers on Exit

Upon return to IMS, all registers must be restored except for register 15, which must contain one of the following return codes. Register 1 contains the message number if register 15 contains a value of 12; otherwise it is ignored. Any other value causes the message to be canceled and the terminal operator to be notified.

Return Code	Meaning
00	Segment is processed normally.
04	Segment is canceled.
08	Message is canceled and the terminal operator is notified.
12	Message is canceled and the message identified by register 1 is sent to the terminal.

Using the Sample Transaction Code (Input) Edit Routine (DFSCSMB0)

Assume a multisegment transaction named ICS. Normally, the first segment of this message contains ICS GN (meaning to get the next segment of a given message), or it contains ICS CAN (meaning to cancel this message). A user-supplied edit routine allows further input flexibility, as shown in the following decision table.

	Message as Received and Edited by IMS	**Message as Reedited User Edit Routine**
First Segment	ICS GN	As received
	ICS	ICS GN
	ICS CAN	Cancel Message
	Any other	Cancel Message
Other Segment	GN	As received
	CAN	Cancel Message
	Any other	Cancel Message

The Transaction Code edit routine allows the input for the ICS GN message segment to be shortened.

Chapter 64. 2972/2980 Input Edit Routine (DFS29800)

This chapter describes the 2972/2980 Input Edit routine.

About This Routine

An Input edit routine is required to perform terminal-related functions inherent in the design of the 2972/2980 General Banking Terminal system. Usage and value of these functional characteristics are installation-oriented, and are therefore not performed by normal IMS procedures. Control is passed to the 2972/2980 Input Edit Routine to process each entered message segment after that message segment has been translated by IMS.

The 2972/2980 Input edit routine must perform the following functions:

1. Determine the IMS destination (SMB or CNT) of messages entered from a 2980 teller or administrative station.
2. Determine end-of-message of multisegment messages (by setting DECCSWST bit 7 to indicate EOM).
3. Reposition the entered data at the beginning of the input buffer for IMS processing the entered segment must be in standard IMS input message format after edit processing; a two-byte length field is followed by the text.

In addition to performing the preceding required functions, the 2972/2980 input edit routine can add input terminal status information to the entered segment, such as the presence or absence of a passbook or auditor key on the input terminal. The input edit routine can initiate retransmission of the last successfully transmitted message to a 2980 logical terminal by a return code to the calling routine.

Table 115 shows the attributes of the 2972/2980 Input Edit exit routine.

Table 115. 2972/2980 Input Edit Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFS29800.
Link editing	Follow the guidelines described in “Link-Editing the Routines” on page 5.
Including the routine	Because it will be called directly by the IMS 2972/2980 device dependent module (DFSDN110), the input edit routine must be link-edited with the IMS control region nucleus.
IMS callable services	To use IMS Callable Services with this routine, you must issue an initialization call (DFSCSII0) to obtain the Callable Service token and a parameter list in which to build the function-specific parameter list for the desired callable service. Use the ECB in register 9 for the DFSCSII0 call. For details on using the initialization call, see “Step 2. Initializing IMS Callable Services (DFSCSII0)” on page 11. This exit is automatically linked to DFSCSI00 by IMS. No additional linking is required to use IMS Callable Services.
Sample routine location	IMS.TMxxSOURCE (member name DFS29800).

Communicating with IMS

Familiarity with IMS terminal handling procedures and control blocks is required for a user to write an Input edit routine to interface with IMS routines in the IMS control region. Examination of these control blocks may be required, but modification of IMS control blocks by a user-written routine seriously endangers the integrity of the entire system.

Contents of Registers on Entry

Upon entry to the exit routine, all registers must be saved using the save area provided. The registers contain the following:

Register	Contents
0	Length of input buffer.
1	Address of the input area.
2	Length of input data. (The length of the area pointed to in register 1.)
7	Address of CTB.
9	Address of CLB.
11	Base of SCD.
13	Address of save area. The first three words must not be changed.
14	Return address to IMS.
15	Entry point of exit routine.

The format of the data contained in the buffer pointed to by register 1 at entry to the 2972/2980 Input edit routine is as follows:

9 Blanks	Terminal Address	Entered Text*

If entry is from a 2980-4, the first byte of the entered text is the teller identification number.

Contents of Registers on Exit

Upon return to IMS, all registers must be restored except for registers 2, 10, and 15, which must contain the following:

Register	Contents
2	Data length after edit (a zero length signifies a no-data segment).
10	The inputting CNT address if a retransmission of the last successfully outputted message is required.
15	One of the following return codes:

Return Code	Meaning
0	Process the entered segment.
4	Resend the last message to the CNT in register 10.

Chapter 65. 4701 Transaction Input Edit Routine (DFS36010)

This chapter describes the 4701 Transaction Input Edit routine.

About This Routine

This exit is provided as a sample routine that appends a blank and the eight-byte node name to a transaction input message. If you have established a naming convention that relates node names to LTERM names, the node name can be used by the MPP to set up the appropriate change call for output.

Table 116 shows the attributes of the 4701 Transaction Input Edit routine.

Table 116. 4701 Transaction Input Edit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFS36010.
Link editing	Follow the guidelines described in “Link-Editing the Routines” on page 5.
Including the routine	No special steps are required to include this routine.
IMS callable services	To use IMS Callable Services with this routine, you must issue an initialization call (DFSCSII0) to obtain the Callable Service token and a parameter list in which to build the function-specific parameter list for the desired callable service. Use the ECB found in register 9 for the DFSCSI00 call. For details on using the initialization call, see “Step 2. Initializing IMS Callable Services (DFSCSII0)” on page 11. This exit is automatically linked to DFSCSI00 by IMS. No additional linking is required to use IMS Callable Services.
Sample routine location	IMS.TMSOURCE (member name DFS36010).

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of Registers on Entry

Upon entry to the exit routine, all registers must be saved using the save area provided. The registers contain the following:

Register	**Contents**
1	Address of the input buffer
7	Address of CTB
9	Address of CLB
11	Address of SCD
13	Address of save area
15	Entry point of exit routine

Contents of Registers on Exit

Upon return to IMS, all registers must be restored except for register 15, which must contain the following:

Register	Contents	
15	Return Code	
	Return Code	**Meaning**
	0	Normal processing

Chapter 66. 7770-3 Input Edit Routine (DFSI7770)

This chapter describes the 7770-3 Input Edit exit routine.

About This Routine

For the 7770-3, an Input edit exit routine has been implemented at the line level from device module DFSDS030. This exit routine is provided primarily for an edit routine to operate conversationally with the line (caller). The exit routine performs basic (no database reference) validity checking of input fields. The 7770-3 has limited error detection. It must also build a transaction, field by field, until enough data has been received and validity checked so that the message (transaction) can be scheduled by IMS. Message text is translated to EBCDIC before the exit routine is activated.

In conjunction with the above concept of input editing, several additional entries and actions have been provided for the Input Edit exit routine to allow the exit routine to be continually aware of the line status from operation to operation.

Table 117 shows the attributes of the 7770-3 Input Edit exit routine.

Table 117. 7770-3 Input Edit Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSI7770.
Link editing	Follow the guidelines described in “Link-Editing the Routines” on page 5.
Including the routine	IMS supplies a basic Input Edit exit routine for the 7770-3 as module DFSI7770 in IMS.TMSOURCE. If you want to use the supplied module, you must assemble it and link edit it into the user library specified in the IMSGEN statement. If you have written your own Input Edit exit routine, that module must be stored in the user library specified in the IMSGEN statement before system definition.
IMS callable services	To use IMS Callable Services with this routine, you must issue an initialization call (DFSCSII0) to obtain the Callable Service token and a parameter list in which to build the function-specific parameter list for the desired callable service. Use the ECB found in register 9 for the DFSCSII0 call. For details on using the initialization call, see “Step 2. Initializing IMS Callable Services (DFSCSII0)” on page 11. This exit is automatically linked to DFSCSI00 by IMS. No additional linking is required to use IMS Callable Services.
Sample routine location	IMS.TMSOURCE (member name DFSI7770).

Communicating with IMS

IMS uses the entry and exit registers to communicate with the routine.

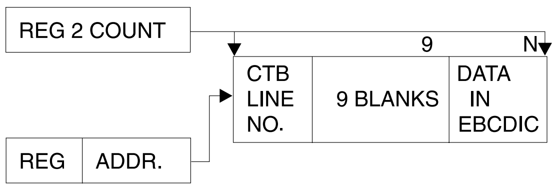
Contents of Registers on Entry

Upon entry to the routine, all registers must be saved using the save area provided. The registers contain the following:

Register	Contents
0	Entry vector value:

Register	Contents	
	Vector	**Value**
	00	Entry is for normal segment read completion from the line (caller).
	04	Reentry for next segment of message after input edit has indicated that it has more segments to send to IMS.
	08	The calling party on the line has hung up.
	12	The line is being stopped or the system is shutting down.
1	Address of the input data/buffer area. If the entry vector is 12, Register 1 is not used.	
2	Length of the input data/buffer area. If the entry vector is 12, Register 2 is not used.	
7	Address of CTB.	
8	Address of CTT.	
9	Address of CLB.	
11	Address of SCD.	
13	Address of save area. The first three words must not be changed.	
14	Return address to IMS.	
15	Entry point of routine.	

The data format on entry is as follows:



Contents of Registers on Exit

Upon return to IMS, all registers must be restored except for registers 0, 1, 2, and 15. The contents of registers 0 and 1 are ignored by IMS. Registers 2 and 15 must contain the following:

Register	Contents	
2	The length of the data now in the input buffer area that was pointed to by Register 1 on entry.	
15	One of the following return codes:	
	Return Code	**Meaning**
	00	The message segment in the input buffer is to be sent to IMS and is the last segment of the message.
	04	The message segment in the input buffer is to be sent to IMS and is not the last segment of the message. The next time the device module is entered for a READ, it enters the edit module with Register 1 pointing to a buffer area, and Register 2 containing the amount of available area contained in the buffer. Register 0 contains the value of 04.
	08	The message in the input buffer is to be sent to the caller followed by a READ. Register 2 must contain the count for the message to be sent to the caller. The message must be in drum address form.
	12	Repeat the last output message for the caller.
	16	The contents of the input buffer should be sent to the caller with a reset to hang up the caller.

Error Conditions

IMS stops the line and generates a message to the master terminal for any one of the following input edit module error conditions:

- The return code from the input edit module exceeds 16.
- The count value returned in register 2 is greater than the available space in the buffer (buffer overrun).
- The input edit module sends a single segment message to IMS after the caller has hung up and indicated that it has more segments to send to IMS.
- The return code from the routine exceeds 8 after entered for disconnect indication.

After the line has been stopped, system messages can still be transmitted to the 7770-3. The Input Edit exit routine is not activated.

Special Conditions

After the edit module has been entered with the 08 entry vector value indicating that the caller has hung up, the edit routine can use return codes 00 and 04 to continue sending data to IMS before IMS is notified of the line drop condition. During this mode of processing, return code 00 indicates the end of input edit control, and indicates that the message should be enqueued for processing. Alternatively, a return code of 08 during this mode causes the message to be canceled and terminates input edit control for this sequence.

IMS does not accept input for conversational transactions if the disconnect occurs during a WRITE operation. The response message from the conversational program is still in the queue, and therefore negates input operations.

No IMS action can be specified if the edit module is entered with entry vector 12. Returned parameters, if any, are not used, because the entry with input vector 12 is an information-only entry. The return code value of 12 or 16 can only be returned after the Input edit routine is entered for a normal READ completion.

The input data can contain one or more of the following special characters:

```
X'00'   For Invalid Input Line Codes
X'16'   For 2721 Cancel Key
X'26'   For EOB (on 2721 also '000' key and '#' Key as EOIs)
X'B0'   For 2721 Verify Key
X'B1'   For 2721 Repeat Key
X'B2'   For 2721 Function 1 (F1) Key
X'B3'   For 2721 Function 2 (F2) Key
X'B4'   For 2721 Function 3 (F3) Key
X'B5'   For 2721 Function 4 (F4) Key
X'B6'   For 2721 Function 5 (F5) Key
X'B7'   For 2721 ID X'19' Code
X'B8'   For 2721 ID X'59' Code
X'B9'   For 2721 ID X'21' Code
X'BA'   For 2721 ID X'61' Code
X'FA'   For 2721 00 Key and for TOUCH-TONE (or equivalent)
        Phone '*'
        Key when working on the ABB' Code Line Interface
```

Chapter 67. 7770-3 Output Edit Routine (DFSO7770)

This chapter describes the 7770-3 Output Edit exit routine.

About This Routine

IMS provides you with the capability of installing a 7770-3 with an installation-tailored vocabulary. IMS cannot, of course, predict this vocabulary. For this reason, the Output Edit exit routine is implemented to allow a user-written module to inspect system messages and terminal-to-terminal message switch messages and convert them, at your discretion, to a message that is compatible with that vocabulary.

Attributes of the Routine

Table 118 shows the attributes of the 7770-3 Output Edit exit routine.

Table 118. 7770-3 Output Edit Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSO7770.
Link editing	Follow the guidelines described in “Link-Editing the Routines” on page 5.
Including the routine	If the IMS-provided Output Edit exit routine is to be used, you must assemble DFS07770 and link-edit it into the user library specified in the IMSGEN statement before system definition. If you are providing your own Output Edit exit routine, the module must be stored in the user library before system definition.
IMS callable services	To use IMS Callable Services with this routine, you must issue an initialization call (DFSCSII0) to obtain the Callable Service token and a parameter list in which to build the function-specific parameter list for the desired callable service. Use the ECB found in register 9 for the DFSCSII0 call. For details on using the initialization call, see “Step 2. Initializing IMS Callable Services (DFSCSII0)” on page 11. This exit routine is automatically linked to DFSCSI00 by IMS. No additional linking is required to use IMS Callable Services.
Sample routine location	IMS.TMSOURCE (member name DFSO7770).

Calling the Routine

The Output Edit exit routine receives control for system messages or message switches. It does not receive control for a message from an application program that is a response to an input transaction.

Communicating with IMS

IMS uses the entry and exit registers to communicate with the Output Edit exit routine.

Contents of Registers on Entry

Upon entry to the Output Edit exit routine, all registers must be saved using the save area provided. The registers contain the following:

Register	Contents
1	Address of the output message segment.
2	Length of the output message segment.
7	Address of CTB.
8	Address of CTT.
9	Address of CLB.
10	Address of CNT.
11	Address of SCD.
13	Address of save area. The first three words must not be changed.
14	Return address to IMS.
15	Entry point of Edit exit routine.

Data Format on Entry

Before control is given to the output edit module, IMS edits the output message into the output buffer until the end of message is reached or the buffer is full. The buffer contains only output message data in EBCDIC.

Contents of Registers on Exit

Upon return to IMS, all registers must be restored except for registers 0, 1, 2, and 15. The contents of registers 0 and 1 are ignored by IMS. Registers 2 and 15 contain the following:

Register	Contents
2	The length of the data now in the output buffer area that was pointed to by Register 1 on entry.
15	One of the following return codes:

Return Code	Meaning
00	No action taken by the output edit. IMS is to continue sending the message and any further segments without routing control to the output edit module.
04	IMS is to send the current contents of the buffer to the line, and the output edit module desires to gain control for any further segments of this message.
08	The contents of the buffer have been changed. IMS is to send what is now in the buffer and ignore (dequeue and not send) any further segments of the message.

Error Conditions

IMS stops the line and generates a message to the master terminal for any one of the following output edit module error conditions:

- The return code from the output edit module exceeds 8.
- The count returned in register 2 is negative or zero.
- The count returned in register 2 is greater than the available buffer space (buffer overrun).

After the line has been stopped, system messages can still be transmitted to the 7770-3. The Output Edit exit routine is not activated.

Special Conditions

The supplied output edit module makes the following assumptions:

- The vocabulary of the 7770-3 contains the phonetic equivalents for the numbers 0 through 9 and that the translate table supplied by the user converts the EBCDIC numbers to their vocabulary equivalents.
- The prefix phrase (in drum address form) to be sent for system messages follows the user translate table, and the orientation phrase and has the form *nppp*, where *n* is a single byte containing the count of the number of drum address bytes (*p*) following. The orientation phrase has the format *nppp*.
- Because of the variable nature of the 7770-3 vocabulary, the system definition utility requires that you supply the output translate table for the 7770-3. It is also your responsibility to provide the required orientation phrase also to be used for system message conversion.

Chapter 68. 7770-3 Sign-on Exit Routine (DFSS7770)

This chapter describes the 7770-3 Sign-on exit routine.

About This Routine

Because the 7770-3 is a switched device and the calling terminal may not be able to generate the alphameric characters required to form an /IAM command to sign on for an LTERM, IMS requires that a Sign-On exit routine be defined at system definition time for the 7770-3 lines in the system. This routine is activated by the 7770-3 device-dependent module any time an input message or message segment is received from the line and a logical connection does not exist. Only one routine can be defined, and it applies to all 7770-3 lines in the system. A minimum user routine should check the validity of input data received from the line, and use the data to develop an /IAM command to be passed to IMS. The user routine gains control before any IMS security checking, validity checking, or editing functions are performed. The message text is in EBCDIC.

The Sign-On exit routine can build an /IAM command in the input buffer, or store a response message in the input buffer. Any response to be sent back to the caller must be in 7770-3 output vocabulary drum address form.

Through return codes to IMS, the Sign-On exit routine can cause the contents of the input buffer to be passed to the system (/IAM command in buffer), or cause the contents of the buffer to be sent to the caller, followed by a READ to allow retry. This routine can also cause the contents of the input buffer to be sent to the caller with a reset to the line to disconnect the caller after the response is sent.

Table 119 shows the attributes of the 7770-3 Sign-on exit routine.

Table 119. 7770-3 Sign-on Exit Routine Attributes

Attribute	Description
IMS environments	DB/DC, DCCTL.
Naming convention	You must name this exit routine DFSS7770.
Link editing	Follow the guidelines described in “Link-Editing the Routines” on page 5. If the supplied module is to be used, you must assemble and link-edit it into the user library specified in the IMSGEN statement before Stage 2 of system definition is executed.
Including the routine	This routine automatically signs the caller on for the INQUIRY LTERM whenever the 7770-3 answers a call and receives data. As supplied, this routine is not apparent to the caller.
IMS callable services	To use IMS Callable Services with this routine, you must issue an initialization call (DFSCSII0) to obtain the Callable Service token and a parameter list in which to build the function-specific parameter list for the desired callable service. Use the ECB found in register 9 for the DFSCSII0 call. For details on using the initialization call, see “Step 2. Initializing IMS Callable Services (DFSCSII0)” on page 11. This exit routine is automatically linked to DFSCSI00 by IMS. No additional linking is required to use IMS Callable Services.
Sample routine location	IMS.TMSOURCE (member name DFSS7770).

Communicating with IMS

IMS uses the entry and exit registers to communicate with the exit routine.

Contents of Registers on Entry

Upon entry to the exit routine, all registers must be saved using the save area provided. The registers contain the following:

Register	Contents
1	Address of input data/buffer area received from the line.
2	Length of the input data/buffer area.
7	Address of CTB.
8	Address of CTT.
9	Address of CLB.
11	Address of SCD.
13	Address of save area. The first three words must not be changed.
14	Return address to IMS.
15	Entry point of exit routine.

The data format at entry and the relationship of registers 1 and 2 to the data are shown in the sample routine in the IMS.TMSOURCE library; member name is DFSS7770.

Contents of Registers on Exit

Upon return to IMS, all registers must be restored except for registers 0, 1, 2, and 15. The contents of registers 0 and 1 are ignored by IMS. Registers 2 and 15 must contain the following:

Register	Contents
2	The length of the data now in the input buffer area that was pointed to by Register 1 on entry.
15	One of the following return codes

Return Code	Meaning
0	Continue input processing with the contents of the input buffer.
4	Send the contents of the input buffer to the caller, followed by a read. Allows retry of sign-on operation.
8	Send the contents of the input buffer to the caller, followed by a disable to disconnect the caller.

For return codes 04 and 08, the contents of the input buffer to be sent to the caller must be in drum address form, because no translation is performed before the data is sent to the caller. You must also determine when a sequence of sign-on attempts should be terminated with a reset operation.

Error Conditions

IMS stops the line and generates a message to the master terminal for either of the following sign-on exit routine error conditions:

- The return code from the sign-on routine exceeds 8.
- The count value returned in Register 2 is greater than the available space in the buffer.

After the line has been stopped, system messages can still be transmitted to the 7770-3. The Sign-On exit routine is not activated.

Part 5. Automated Operator Program Interface

This section documents General-Use Programming Interface and Associated Guidance Information. See “Notices” on page xxi to understand this information classification.

Chapter 69. Automated Operator (AO) Application Program (GU, GN, CMD, and GCMD Calls)

There are two types of automated operator (AO) application programs you can write. The AO application described in this chapter is called a type 1 AO application. *Type 1 AO applications* issue DL/I GU, GN, CMD, and GCMD calls in the DB/DC and DCCTL environments to:

- Retrieve messages from AO exit routine DFSAOUE0 (GU, and GN calls).
- Issue a subset of IMS operator commands (CMD call).
- Retrieve command responses for commands issued on the CMD call (GCMD call).

Related Reading:

- *Type 2 AO applications* are described in "Chapter 70. Automated Operator (AO) Application Program (GMSG, ICMD, and RCMD Calls)" on page 445. They use GMSG, ICMD, and RCMD calls and are applicable in the DB/DC, DCCTL, and DBCTL environments.
- For an overview and comparison of the two types of AO applications, see *IMS/ESA Operations Guide*.

About This Application

The Automated Operator Interface (AOI) consists of several IMS functions that allow installations to better monitor and control IMS activities. These functions can be used separately or in conjunction with user-written programs and the master terminal operator.

For example, one function includes a system definition option and an operator command that gives the master terminal operator more control of the command messages sent to the secondary master terminal.

An AO application can issue a subset of IMS commands normally reserved for the master terminal operator and receive responses to those commands by using the following DL/I calls:

- CMD (to issue commands and receive the first command response segment)
- GCMD (to receive subsequent command response segments)

If there are any response segments, the first is returned to the user's I/O work area. Subsequent segments are retrieved with GCMD. GCMD, similar to a get next (GN), places subsequent command response segments in the user's I/O work area.

The AO application receives control from IMS whenever a message with the correct transaction code has been queued for processing.

Figure 23 on page 436 shows an AO application issuing a command with the CMD call **1** and receiving the initial response segment to the command **2**. The AO application uses the GCMD call **3** to obtain additional response segments **4**, if they exist.

Another AOI function lets an AO application retrieve messages. GU and GN calls retrieve messages. The retrieved message can be from either:

- A terminal from which the transaction was entered

Type 1 AO

- AO exit routine DFSAOUE0

An AO application using this function typically retrieves messages from the AO exit routine and then takes specific actions based on the content of the message.

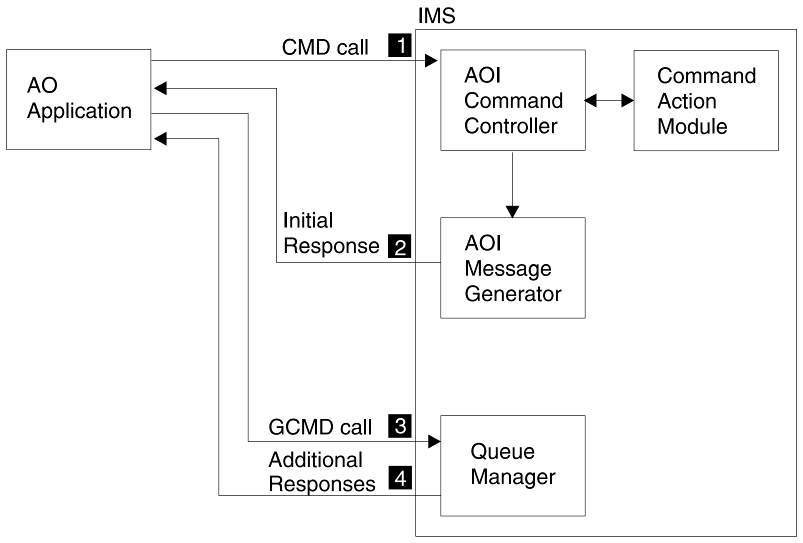


Figure 23. AO Application Processing

Supported Application Program Environments

Table 120 shows, by IMS environment, the types of AO applications that can issue GU, GN, CMD, and GCMD calls.

Table 120. GU, GN, CMD, and GCMD Call Support by Application Region Type

Application Region Type	**IMS Environment**		
	DBCTL	***DB/DC***	***DCCTL***
DRA thread	No	No	N/A
BMP (nonmessage-driven)	No	No	No
BMP (message-driven)	N/A	Yes	Yes
MPP	N/A	Yes	Yes
IFP	N/A	No	No

Writing AO Applications for Shared Queues

Some transactions must reside on the same IMS subsystem as DFSAOUE0 to process correctly. If your installation uses shared queues, define these local transactions as SERIAL to guarantee that they are processed on the local IMS subsystem. A transaction that is not defined as SERIAL can be processed on any IMS subsystem that has that transaction defined.

Using the Security Maintenance Utility

Use the Security Maintenance Utility (SMU) to define the authorized application programs by coding the TRANCMDS keyword on the SECURITY macro. The SECCNT and MACLIB keywords on the IMSGEN macro must also be coded. The COPYLOG keyword must be coded on the COMM macro if the secondary logging feature is desired.

Related Reading: For details on coding the IMSGEN and SECURITY macros and using SMU to authorize transactions, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Using SMU, you can:
- Specify which AO applications can issue commands using the CMD call.
- Specify which commands can be issued by an AO application.
- Specify that an AO application can issue all commands allowed as indicated by the system-definition-produced list of allowed commands.

Only those AO applications specified to SMU are allowed to enter commands. Also, SMU only applies to AO applications issuing commands using the CMD call. If an AO application is not specified to SMU (in other words, if default security is in effect), it cannot issue commands.

Related Reading: For more information on SMU, see *IMS/ESA Administration Guide: System*.

Issuing Commands from an AO Application

See the *IMS/ESA Operator’s Reference* for a list of commands that can be issued from an AO application using CMD. To receive responses to these commands, use the GCMD call.

Format of Commands

The syntax and synonyms for commands are the same whether the command is entered using a CMD call or from a terminal. All messages returned in response to a CMD call or a command entered from the terminal are the same.

Related Reading:
- All commands are documented in *IMS/ESA Operator’s Reference*.
- All messages are documented in *IMS/ESA Messages and Codes*.

The general format of your I/O work area on a CMD call is:

LLZZ/*verb* **KEYWORD1 P1 KEYWORD2 P2, P3.** *Comments*

LL	Two-byte field containing the length of the command text, including **LLZZ**
ZZ	Two-byte field reserved for IMS
/	Indicates that an IMS command follows
verb	The command you issued
KEYWORD*x*	Keywords that apply to the command you issued
P*x*	Parameters for the keywords you specified

. (period)	End of the command

The length of a command is limited by the size of the I/O area; the size is specified in the IOASIZE parameter in the PSBGEN macro during PCB generation. LL is the length of the command text. The size of the I/O area is the length of the actual command text, plus 4 bytes for LLZZ. The minimum size of the I/O work area is 132 bytes. The fifth byte must be a slash (/), and the verb must follow immediately. The /BROADCAST and /LOOPTEST commands must have a period between the command segment and text segment, and must be preceded by an LLZZ field that includes the size of the text. Comments can be added by placing a period (.) after the last parameter.

When issuing the /SSR command, do not code an end-of-message indicator (period) as shown in the *IMS/ESA Operator's Reference*. If a period is used, it is considered part of the text.

Responses to Commands

The execution of a command with a CMD call might result in a response that comprises one or more segments. The response is similar to that received by a terminal operator upon entering the same command. The maximum size response returned by IMS to the AO application is 132 bytes (including the 4-byte LLZZ field).

DFS messages that are the initial response to the command will be time-stamped even if the NOTIMESTP keyword was coded on the COMM macro.

No Response Segments

This condition is indicated by a PCB status of blanks, showing that the execution of the command was successful or is in progress and would have resulted in one of the following command responses had the command been entered from a terminal:

```
DFS058 XXX COMMAND COMPLETED (No EXCEPT phrase)
DFS058 XXX COMMAND IN PROGRESS.
```

where XXX is the command issued by the automated operator.

DFS058 COMMAND COMPLETED indicates that no exceptional conditions are reported.

When DFS058 COMMAND IN PROGRESS is issued, one or more additional messages are issued to indicate whether the command was successful. These messages are sent to the master terminal rather than to AOI. Commands such as /DBDDUMP, /DBRECOVERY, /START DATABASE, /STOP DATABASE, and /SSR can result in message DFS058, followed by one or more asynchronous messages. Asynchronous messages are not returned as responses to the AO application.

Response Segments

Response segments are identical to the response to a command entered by the operator, with preceding LLZZ and carriage control (optional) and trailing carriage controls (optional). A command might have executed successfully even though a nonblank PCB status code is returned. The status code CC indicates that one or more response segments have been produced, which occurs when a command partially executes. For example, the EXCEPT condition of this message might be returned as multisegments.

Synchronization Point Processing

When the AO application regains control after a CMD call, IMS produces an initial response to the command. Command processing might be incomplete because additional IMS processing initiated by the command is still in progress.

Command responses that are not retrieved by a GCMD call are discarded at a synchronization point or by another CMD call.

A *synchronization point* is reached when any of the following events occur:

- The AO application requests the next input message with a GU call, assuming the program is defined as MODE=SINGLE on the TRANSACT macro statement.
- The program makes a CHKP call.
- The program makes a SYNC call.
- The program terminates.

Therefore, when IMS is restarted or resets to an earlier checkpoint, it is possible that a previously-issued command will be rescheduled for execution. An input message, generated by an AO exit routine and destined for an AO application, could be rescheduled although the resources are no longer held by the AO application. PCB status codes indicate more precisely which condition, or combination of conditions, exists.

/DISPLAY Command

When a `/DISPLAY` command is entered with a CMD call, a command response segment is always produced. Normally, the response is a multisegment message. Each segment contains EBCDIC and line control characters. Each segment varies in length with a maximum of 132 but usually less than 80 characters.

Output from the `/DISPLAY` command includes a format identifier (FID) on each output segment. `/DISPLAY` commands generally return multiple lines with different formats to the AO application. Your AO application uses the FID to determine what type of line it is processing. A DFS message might be produced with no FID when there is a syntax error.

The FID is 3 bytes of EBCDIC characters. It appears at the beginning of each line, after the LLZZ and any optional carriage control character. The format of the FID is:

FID character 1
Specific `/DISPLAY` command class (A-Z) as follows:

FID characters 2-3
Type of output line

Listed below are the values you can receive in FID characters 1-3 when issuing `/DISPLAY` using an ICMD.

FID char 1 Defines specific `/DISPLAY` command:

A	`/DISPLAY ACTIVE`, `/DIS AOITOKEN`, `/DISPLAY APPC`
B	`/DISPLAY PSB`
C	`/DISPLAY CONVERSATION`
D	`/DISPLAY AREA`, `/DISPLAY DATABASE`, `/DISPLAY DESCRIPTOR`
E	`/DISPLAY MSNAME`

F	`/DISPLAY LTERM`
G	`/DISPLAY SHUTDOWN STATUS`
H	`/DISPLAY ASMT LINE/PTERM/LTERM/NODE/USER`, `/DISPLAY HSB`
I	`/DISPLAY ASMT LINK/MSPLINK/SYSID/MSNAME`
J	`/DISPLAY DBD`
K	`/DISPLAY LINK`
L	`/DISPLAY LINE`, `/DISPLAY LUNAME`, `/DISPLAY PTERM`
M	`/DISPLAY MASTER (/RDISPLAY)`
N	`/DISPLAY NODE`
P	`/DISPLAY PROGRAM`
Q	`/DISPLAY Q`
R	`/DISPLAY RTCODE`
S	`/DISPLAY STATUS`
T	`/DISPLAY TRANSACTION`, `/DISPLAY TIMEOVER`, `/DISPLAY SYSID TRANSACTION`
U	`/DISPLAY USER`, `/DISPLAY TRACE`
V	`/DISPLAY CCTL`, `/DISPLAY OASN SUBSYS`, `/DISPLAY SUBSYS`
W	`/DISPLAY OLDS`
X	Timestamp, `/DISPLAY POOL DSECTS`
Y	`/DISPLAY MODIFY`
Z	`/DISPLAY HSSP`

FID chars 2-3 Defines type of output line:

00-49	Data line
50-69	Message line
70-89	Heading line
90-98	Reserved
99	Time stamp line

The response to the `/DISPLAY POOL` command does not include a FID. You can get similar information about a buffer pool using a DL/I STAT call, although the STAT call returns no Fast Path information.

Getting Messages from an AO Exit Routine or a Terminal

Your AO application receives control whenever a message with the appropriate transaction code is queued for processing. Your AO application retrieves messages from an AO exit routine or a terminal using the GU call.

Cross-Reference of Commands and Command Response Messages

Table 121 associates IMS commands used in an AO application and the resulting DFS response messages. You'll want to refer to Table 121 when writing your AO application to know what error messages to expect and be able to plan for them in your application.

Most of the messages in Table 121 are error messages that the AO application receives when an IMS command executes improperly. When an IMS command **does** execute properly, the program does not receive message DFS058 confirming that the command completed successfully.

Exception: This is true for all commands except the /DISPLAY, /RDISPLAY, and the /RMxxxxxx commands. When /DISPLAY, /RDISPLAY, or /RMxxxxxx is successfully completed, the AO application receives the data it requested to be displayed rather than message DFS058.

Related Reading: If you want detailed descriptions of the messages in Table 121, see "DFS Messages," in *IMS/ESA Messages and Codes*.

Table 121. IMS Commands and DFS Response Messages

Command	DFS Response Message
/ACTIVATE	107 127 158 163 181 182 2103
/ALLOCATE	107 108 121 127 158 163 182 216 1950 1952 1953 2038 3110
/ASSIGN	107 119 124 127 131 138 139 147 150 151 152 157 158 161 163 164 182 201 210 211 212 213 214 232 243 289 794 795 2034 2035 2036 2104 2105 3102 3103 3104 3105 3108 3115 3632 3636 3637 3638
/BROADCAST	099 100 102 113 115 127 147 158 163 182 2105 3633
/CHANGE	107 108 121 125 126 127 130 163 181 182 216 696 1951 1953 2038 2105 2231 2232 3458 3619 3630 3632 3633 3639 3805 3811 3812 3817 3818 3819 3826 3827 3832 3833
/CHECKPOINT	107 127 130 140 141 142 163 182 2038 2717
/CLSDST/OPNDST	107 121 127 128 130 140 147 163 181 182 2037 2081 2103 2108 2109 2477 3101 3105 3106 3107 3108 3110 3111 3112 3113 3114 3116 3630 3633 3634 3639 3862
/COMPT	107 127 163 182 2103 2104 2254 2255 2256 2257 2261 3106 3108
/DBDUMP	107 127 130 132 140 141 142 160 163 174 175 182 2038 2717 3316 3319
/DBRECOVERY	107 121 127 130 132 140 141 142 158 160 163 174 175 178 182 2038 2537 2717 3316 3319 3327
/DELETE	102 106 107 109 113 114 115 116 117 123 127 128 130 146 150 158 163 2103 2104 2105 3654
/DEQUEUE	107 113 114 115 121 127 128 130 150 151 152 158 163 182 189 216 236 237 238 239 240 1191 1192 1952 1953 2038 2103 2104 2153 2183 2185 3104 3106 3107 3108 3110 3117 3824
All DISPLAYs	074 107 127 130 163 182 1924 2038 2093
/DISPLAY APPC	127 163 182 1953

Table 121. IMS Commands and DFS Response Messages (continued)

Command	DFS Response Message
/DISPLAY ASSIGNMENT SYSID MSNAME MSPLINK LINK	147 147 147 147
/DISPLAY AREA	116 132

/DISPLAY ASSIGNMENT NODE
LTERM
LINE-PTE
USER
	182 2103 3106 3630 3639
	114
	113 115 150
	3108

/DISPLAY CCTL	182 216
/DISPLAY CONVERSATION	099 113 115 163 182 216 2103 3108
/DISPLAY DATABASE	116 132 3653
/DISPLAY DBD	182
/DISPLAY DESCRIPTOR	121 127 163 182 1953
/DISPLAY HSB	182 3813
/DISPLAY LINE-PTE	113 115 150 182 216
/DISPLAY LINK	147 2038
/DISPLAY LTERM	114 216 2038
/DISPLAY LUNAME	107 121 127 130 158 163 182 1953 1954 3110
/DISPLAY MASTER	
/DISPLAY MODIFY	163 182 237 3431 3443 3451 3452 3455 3460 3461
/DISPLAY MSNAME	114 216 2038
/DISPLAY NODE	182 216 2103 3106 3114 3630 3639 3831
/DISPLAY OASN SUBSYS	147 182 216 3601
/DISPLAY OLDS	182 216 2038 3828
/DISPLAY PROGRAM	117
/DISPLAY PSB	147 182
/DISPLAY Q	127 147 156 163 170 216 2038
/DISPLAY RTCODE	147
/DISPLAY SHUTDOWN STATUS	134 198
/DISPLAY SUBSYS	147 182 216 3601
/DISPLAY SYSID TRAN	147
/DISPLAY TRANSACTION	146 216
/DISPLAY USER	3108 3653
/END	113 115 150 152 158 181 182 189 285 2103 2104 3106 3108 3635 3654
/EXCLUSIVE	113 115 150 152 158 181 182 189 241 2103 2104 3106 3108 3635 3654
/EXIT	107 180 181 182 183 184 189 576 577 2103 2104 2105 3106 3108 3635 3655
/FORMAT	070 182 244 3824
/IDLE	107 127 130 134 147 163 181 216 2105 2109 3633
/LOCK	062 107 114 116 117 127 146 158 162 163 182 2038 3250

Table 121. IMS Commands and DFS Response Messages (continued)

Command	DFS Response Message
/LOG	
/LOOPTEST	113 115 143 150 152 181 195 196 203
/MONITOR	099 107 113 115 130 140 150 163 182 216
/MSASSIGN	107 127 147 163 164 182 2038 2151 2152 2153 2154 2155 2156 2157 2233 2240 2244 2252 2259 2260 3200 3214
/PSTOP	099 107 113 114 115 117 130 140 146 147 150 156 163 181 182 216 232 2152 2270 2272 2273 2297 3204 3630 3633 3824
/PURGE	099 107 108 113 114 115 117 127 130 140 146 150 156 163 182 216 1953 2038 3630 3633 3824
/QUIESCE	107 130 140 158 163 181 182 216 2103 3106 3630 3632 3633 3824
/RDISPLAY	
/RM	181 182 2038 3322
/RSTART	099 107 113 115 130 140 147 150 163 181 182 216 2103 2152 3104 3106 3108 3204 3630 3633 3639 3824
/SECURE	107 127 163 182 1953
/SMCOPY	165 181 182
/SSR	182 696 2038 3601 3609 3610 3618
/START	099 107 108 113 114 115 117 127 130 132 140 146 150 156 158 163 170 175 176 178 179 182 216 232 696 1953 2010 2020 2021 2022 2023 2024 2025 2038 2103 2105 2475 3104 3106 3108 3110 3315 3316 3319 3601 3604 3609 3630 3633 3639 3798 3799 3805 3806 3807 3808 3809 3816 3817 3824 3829 3843
/STOP	099 107 108 113 114 115 117 127 130 132 140 146 150 156 158 163 170 175 176 178 179 182 216 232 696 1953 2010 2020 2021 2022 2023 2024 2025 2038 2103 2105 2109 2475 3104 3106 3108 3110 3315 3316 3319 3601 3604 3630 3633 3639 3805 3806 3808 3824 3825 3829
/SWITCH	107 127 140 163 696 2038 3810 3813 3814 3824
/TEST	113 115 143 150 152 158 181 182 189 196 282 283 284 2103 2104 2158 3106 3108 3635 3654
/TRACE	107 108 113 130 147 150 158 163 182 216 237 279 280 281 314 775 1953 2028 2029 2030 2031 2032 2038 2093 2103 2104 2155 2460 3106 3108 3110 3630 3633 3639 3813
/UNLOCK	062 107 114 116 117 127 146 158 162 163 182 2038 3250 3813
/VUNLOAD	107 130 140 163 182 216

Using the Sample AO Application (UETRANS)

A sample AO application is available from the IMS library (IMS.TMSOURCE, member name UETRANS).

The sample shows use of AOI by an AO application and an associated AO exit routine. The AO application is written in PL/I for the Optimizing Compiler.

Type 1 AO

The sample AO application:

- Accepts only messages originating from the exit routine (indicated by a `CG` PCB status code).
- Upon receipt of a DFS994 message, looks for the character string `EMERGENCY`. If found, issues the command `/RSTART LINE ALL`. Otherwise, issues the following commands:

  ```
  /RSTART LINE ALL
  /START DATABASE ALL
  /START TRAN ALL
  ```
- Upon receipt of a terminal error message, issues the command `/RDISPLAY` to determine on which line and PTERM is the master terminal. If the error occurred on that line, issues a `/DISPLAY LINE ALL` to determine the first available started line, and reassigns the master terminal to that line and PTERM.
- Upon receipt of an `/ASSIGN` command, if the response is a DFS119 message, reissues the command until it is successful. If the response is a DFS138 message, it restarts the line with a `/RSTART` command and reissues the `/ASSIGN` command. It reissues the `/ASSIGN` command a maximum of 20 times; if the command is not successfully completed after the 20th attempt, an appropriate message is issued.

The associated AO exit routine is assumed to:

- Pass all DFS994 (checkpoint identifier) messages to the AO application.
- Pass all terminal error messages to the AO application. The line number of the terminal in error is placed in the last 20 bytes of the buffer. Message numbers passed are DFS001, DFS025, DFS026, DFS027, DFS029, DFS072, DFS250, DFS251, DFS253, DFS260, DFS269, DFS973, DFS998.
- Pass all `/ASSIGN` commands that receive a DFS119 or DFS138 message to the AO application. These messages indicate that either the LTERM is busy or the line is not started and cannot be reassigned.
- Ignore all other messages.

Chapter 70. Automated Operator (AO) Application Program (GMSG, ICMD, and RCMD Calls)

There are two types of automated operator (AO) application programs you can write. The AO application described in this chapter is called a type 2 AO application. It is applicable in the DB/DC, DCCTL, and DBCTL environments. *Type 2 AO applications* can issue DL/I GMSG, ICMD, and RCMD calls to:

- Retrieve messages from AO exit routine DFSAOE00 (GMSG call).
- Issue a subset of IMS operator commands (ICMD call).
- Retrieve command responses for commands issued on the ICMD call (RCMD calls).

Related Reading:

- Type 1 AO applications are described in "Chapter 69. Automated Operator (AO) Application Program (GU, GN, CMD, and GCMD Calls)" on page 435. They use GU, GN, CMD, and GCMD calls and are applicable in DB/DC and DCCTL environments.
- For an overview and comparison of the two types of AO applications, see *IMS/ESA Operations Guide*.

About This Application

This automated operator (AO) application can help you monitor and control IMS activities. You can use it separately or with AO exit routine DFSAOE00.

This AO application can:

- Issue DL/I calls to retrieve messages from DFSAOE00 (GMSG call).
- Issue a subset of IMS operator commands (ICMD call).
- Retrieve responses to those commands from DFSAOE00 (RCMD call).

Related Reading: For detailed information on the GMSG, ICMD, and RCMD calls, see *IMS/ESA Application Programming: Database Manager*or *IMS/ESA Application Programming: Transaction Manager*.

Retrieving Messages (GMSG Call)

When AO exit routine DFSAOE00 is used to route messages to an AO application, the IMS message queues are not used. Instead, applications that cannot access the message queues or that do not wish to use the message queues retrieve the messages from an AO exit routine.

The AO application retrieves messages using the GMSG call. GMSG associates an AOI token name with a message. The application passes an 8-byte AOI token to IMS, and IMS returns to the application a message associated with the AOI token. By using different AOI tokens, DFSAOE00 can direct messages to different AO applications.

The GMSG call, specifying an 8-byte token, retrieves the first segment of the message associated with the token. Subsequent GMSG calls, with no token specified, retrieve the second through n segments of a multisegment message. If an application needs all segments of a multisegment message, it must issue GMSG until all segments are retrieved before issuing another GMSG call with a token specified. IMS discards all remaining message segments from the previous GMSG call when a new GMSG call specifying an AOI token is issued from the same AO application.

An application can specify a wait on the GMSG call. If there are no messages for the specified AOI token, the application is put into a wait state until a message is available. The decision to wait is specified by the application, unlike a WFI (wait for input) transaction where the wait is specified in the transaction definition.

Defining the AOI Token

You define the AOI token value. You also decide how the AO application gets the AOI token value. Some possible methods for doing these follow:

- The token could be a hard-coded value understood by both DFSAOE00 and the AO application.
- A naming convention could be established that relates the name of an AOI token to an AO application. For example, an AO application with the name ABCAOI02 might use AOI token AOI02nnn.
- The AO application could issue an ICMD '/LOG data ' call with the AOI token in the data. The application would then issue a GMSG call with WAITAOI specified for the AOI token named in the LOG call.

  DFSAOE00 will see the ICMD and can parse the data in the LOG command to get the AOI token name and any other information in the data. DFSAOE00 inserts messages to the AOI token name; the messages are then returned to your AO application when the GMSG call is issued.
- DFSAOE00 could create a dynamic AOI token when it processes an IMS message. Then DFSAOE00 could issue an SVC 34 to start a BMP. The START command can override the APARM parameter on the EXEC statement. The APARM parameter is a way of passing information to the AO application. To retrieve the value specified on the APARM parameter, the AO application can issue the DL/I INQY call.
- DFSAOE00 could store an AOI token in an area that both DFSAOE00 and the AO application have access to. The AO application, when it starts, could get the AOI token from this area.

Issuing Commands and Retrieving Command Responses (ICMD and RCMD Calls)

ICMD issues an IMS command and retrieves the first command response segment if one exists. RCMD retrieves subsequent command response segments. When the response to an ICMD call results in a multisegment message, you may or may not want to retrieve all message segments. If you need all message segments, you must issue RCMDs until all segments are retrieved before issuing another ICMD call. IMS discards all remaining message segments from the previous ICMD call when a new ICMD is issued from the same AO application.

Restriction

All messages queued to an AO application by DFSAOE00 remain in the IMS subsystem to which they are queued. Only AO applications in the local subsystem can issue a DL/I GMSG call to access the queued messages. Because these messages are not written to a shared queue structure, applications on other IMS subsystems cannot access them.

Supported Application Program Environments

Table 122 shows, by IMS environment, the types of AO applications that can issue GMSG, ICMD, and RCMD. In addition, these calls are supported from a CPI-C driven program.

Table 122. GMSG, ICMD, and RCMD Call Support by Application Region Type

	IMS Environment		
Application Region Type	**DBCTL**	**DB/DC**	**DCCTL**
DRA thread	Yes	Yes	N/A
BMP (nonmessage-driven)	Yes	Yes	Yes
BMP (message-driven)	N/A	Yes	Yes
MPP	N/A	Yes	Yes
IFP	N/A	Yes	Yes

Establishing Security

Security consists of controlling which applications can issue the ICMD call and which commands they can specify using ICMD.

ICMD security is implemented using RACF (or equivalent), the Command Authorization Exit Routine (DFSCCMD0), or both. To specify one or both methods of securing ICMD, use the AOIS parameter in the DBC, DCC, and IMS procedures.

Related Reading: See the documentation for these procedures in *IMS/ESA Installation Volume 2: System Definition and Tailoring*for more information on the AOIS parameter.

You need to use the following RACF commands to implement security for AO applications:

ADDGROUP
ADDUSER
PERMIT
RDEFINE

Related Reading: See *IMS/ESA Administration Guide: System* and the RACF library for more detailed information on these topics.

Issuing Commands from an AO Application

See the *IMS/ESA Operator's Reference* for a list of commands that can be issued from an AO application using ICMD. To receive responses to these commands, use the GMSG call.

Format of Commands

The syntax and synonyms for commands are the same whether the command is entered using an ICMD call or from a terminal.

Related Reading: All commands are documented in *IMS/ESA Operator's Reference*.

Type 2 AO

All messages returned in response to an ICMD call or a command entered from the terminal are the same. All messages are documented in *IMS/ESA Messages and Codes*.

The general format of your I/O work area on an ICMD call is:

LLZZ/*verb* **KEYWORD1 P1 KEYWORD2 P2, P3.** *Comments*

LL	Two-byte field containing the length of the command text, including **LLZZ**.
ZZ	Two-byte field reserved for IMS.
/ or CRC	Indicates an IMS command follows. CRC (command recognition character) rather than a slash (/) is used in the DBCTL environment.
verb	The command you issued.
KEYWORD*x*	Keywords that apply to the command you issued.
P*x*	Parameters for the keywords you specified.
. (period)	End of the command.

The length of a command is limited by the size of the I/O area; the size is specified in the IOASIZE parameter in the PSBGEN macro during PCB generation. LL is the length of the command text. The size of the I/O area is the length of the actual command text, plus 4 bytes for LLZZ. The minimum size of the I/O work area is 132 bytes.

The fifth byte must be a slash (/) (or CRC for DBCTL), and the verb must follow immediately. The `/BROADCAST` and `/LOOPTEST` commands must have a period between the command segment and text segment, and must be preceded by an LLZZ field that includes the size of the text. Comments can be added by placing a period (.) after the last parameter.

When issuing the `/SSR` command, do not code an end-of-message indicator (period) as shown in the *IMS/ESA Operator's Reference*. If a period is used, it is considered part of the text.

Responses to Commands

The execution of a command with an ICMD call might result in a response that comprises one or more segments. The response is similar to that received by a terminal operator upon entering the same command.

The maximum size of the response returned by IMS to the AO application is 132 bytes (including the 4-byte LLZZ field). DFS messages that are the initial response to the command are time-stamped even if the NOTIMESTP keyword is coded on the COMM macro.

No Response Segments

This condition is indicated when AIB return and reason codes are zero and the length of the message returned is zero. These indicators tell you execution of the command was successful or is in progress and would have resulted in one of the following command responses had the command been entered from a terminal:

```
DFS058 XXX COMMAND COMPLETED (No EXCEPT phrase)
DFS058 XXX COMMAND IN PROGRESS.
```

where `XXX` is the command issued by the automated operator.

When `DFS058 COMMAND COMPLETED` is issued, no exceptional conditions are reported.

When `DFS058 COMMAND IN PROGRESS` is issued, one or more additional messages are issued to indicate whether the command was successful. These messages are sent to the master terminal rather than to AOI. Commands such as `/DBDDUMP`, `/DBRECOVERY`, `/START DATABASE`, `/STOP DATABASE`, and `/SSR` can result in message DFS058, followed by one or more asynchronous messages. Asynchronous messages are not returned as responses to the AO application.

Response Segments

Response segments are identical to the response to a command entered by the operator, with preceding LLZZ and carriage control (optional) and trailing carriage controls (optional). Command responses not retrieved by an RCMD call are discarded on the next ICMD call.

/DISPLAY Command

When a `/DISPLAY` command is entered with an ICMD call, a command response segment is always produced. Normally, the response is a multisegment message. Each segment contains EBCDIC and line control characters. Each segment varies in length with a maximum of 132 but usually less than 80 characters.

Output from the `/DISPLAY` command includes a format identifier (FID) on each output segment. `/DISPLAY` commands generally return multiple lines with different formats to the AO application. Your AO application uses the FID to determine what type of line it is processing. A DFS message might be produced with no FID when there is a syntax error.

The FID is 3 bytes of EBCDIC characters. It appears at the beginning of each line, after the LLZZ and any optional carriage control character. The format of the FID is:

FID character 1	Specific `/DISPLAY` command class (A-Z)
FID characters 2-3	Type of output line

Listed below are the values you can receive in FID characters 1-3 when issuing `/DISPLAY` using an ICMD.

FID char 1 Defines specific `/DISPLAY` command:

A	`/DISPLAY ACTIVE, /DIS AOITOKEN, /DISPLAY APPC`
B	`/DISPLAY PSB`
C	`/DISPLAY CONVERSATION`
D	`/DISPLAY AREA, /DISPLAY DATABASE, /DISPLAY DESCRIPTOR`
E	`/DISPLAY MSNAME`
F	`/DISPLAY LTERM`
G	`/DISPLAY SHUTDOWN STATUS`
H	`/DISPLAY ASMT LINE/PTERM/LTERM/NODE/USER, /DISPLAY HSB`
I	`/DISPLAY ASMT LINK/MSPLINK/SYSID/MSNAME`
J	`/DISPLAY DBD`
K	`/DISPLAY LINK`

L	`/DISPLAY LINE, /DISPLAY LUNAME, /DISPLAY PTERM`
M	`/DISPLAY MASTER (/RDISPLAY)`
N	`/DISPLAY NODE`
P	`/DISPLAY PROGRAM`
Q	`/DISPLAY Q`
R	`/DISPLAY RTCODE`
S	`/DISPLAY STATUS`
T	`/DISPLAY TRANSACTION, /DISPLAY TIMEOVER, /DISPLAY SYSID TRANSACTION`
U	`/DISPLAY USER, /DISPLAY TRACE`
V	`/DISPLAY CCTL, /DISPLAY OASN SUBSYS, /DISPLAY SUBSYS`
W	`/DISPLAY OLDS`
X	Timestamp, `/DISPLAY POOL DSECTS`
Y	`/DISPLAY MODIFY`
Z	`/DISPLAY HSSP`

FID chars 2-3 Defines type of output line:

00-49	Data line
50-69	Message line
70-89	Heading line
90-98	Reserved
99	Timestamp line

The response to the `/DISPLAY POOL` command does not include a FID. You can get similar information about a buffer pool using a DL/I STAT call, although the STAT call returns no Fast Path information.

Cross-Reference of Commands and Command Response Messages

Table 123 on page 451 associates IMS commands used in an AO application and the resulting DFS response messages. You'll want to refer to Table 123 on page 451 when writing your AO application to know what error messages to expect and to be able to plan for them in your application.

Most of the messages in Table 123 on page 451 are error messages that the AO application receives when an IMS command executes correctly. When an IMS command executes correctly, the program **does not** receive message DFS058 confirming that the command completed successfully.

This is true for all commands except the `/DISPLAY`, `/RDISPLAY`, and the `/RMxxxxx` commands. When `/DISPLAY`, `/RDISPLAY`, or `/RMxxxx` is successfully completed, the AO application receives the data it requested to be displayed rather than message DFS058.

Related Reading: For detailed descriptions of the messages in Table 123 on page 451, see “DFS Messages” in *IMS/ESA Messages and Codes*.

Table 123. IMS Commands and DFS Response Messages

Command	DFS Response Message
/ACTIVATE	107 127 158 163 181 182 2103
/ALLOCATE	107 108 121 127 158 163 182 216 1950 1952 1953 2038 3110
/ASSIGN	107 119 124 127 131 138 139 147 150 151 152 157 158 161 163 164 182 201 210 211 212 213 214 232 243 289 794 795 2034 2035 2036 2104 2105 3102 3103 3104 3105 3108 3115 3632 3636 3637 3638
/BROADCAST	099 100 102 113 115 127 147 158 163 182 2105 3633
/CHANGE	107 108 121 125 126 127 130 163 181 182 216 696 1951 1953 2038 2105 2231 2232 3458 3619 3630 3632 3633 3639 3805 3811 3812 3817 3818 3819 3826 3827 3832 3833
/CHECKPOINT	107 127 130 140 141 142 163 182 2038 2717
/CLSDST/OPNDST	107 121 127 128 130 140 147 163 181 182 2037 2081 2103 2108 2109 2477 3101 3105 3106 3107 3108 3110 3111 3112 3113 3114 3116 3630 3633 3634 3639 3862
/COMPT	107 127 163 182 2103 2104 2254 2255 2256 2257 2261 3106 3108
/DBDUMP	107 127 130 132 140 141 142 160 163 174 175 182 2038 2717 3316 3319
/DBRECOVERY	107 121 127 130 132 140 141 142 158 160 163 174 175 178 182 2038 2537 2717 3316 3319 3327
/DELETE	102 106 107 109 113 114 115 116 117 123 127 128 130 146 150 158 163 2103 2104 2105 3654
/DEQUEUE	107 113 114 115 121 127 128 130 150 151 152 158 163 182 189 216 236 237 238 239 240 1191 1192 1952 1953 2038 2103 2104 2153 2183 2185 3104 3106 3107 3108 3110 3117 3824
All DISPLAYs	074 107 127 130 163 182 1924 2038 2093
/DISPLAY APPC	127 163 182 1953
/DISPLAY ASSIGNMENT SYSID MSNAME MSPLINK LINK	147 147 147 147
/DISPLAY AREA	116 132

/DISPLAY ASSIGNMENT NODE
LTERM
LINE-PTE
USER
	182 2103 3106 3630 3639
	114
	113 115 150
	3108

/DISPLAY CCTL	182 216
/DISPLAY CONVERSATION	099 113 115 163 182 216 2103 3108
/DISPLAY DATABASE	116 132 3653
/DISPLAY DBD	182
/DISPLAY DESCRIPTOR	121 127 163 182 1953
/DISPLAY HSB	182 3813
/DISPLAY LINE-PTE	113 115 150 182 216
/DISPLAY LINK	147 2038
/DISPLAY LTERM	114 216 2038
/DISPLAY LUNAME	107 121 127 130 158 163 182 1953 1954 3110

Table 123. IMS Commands and DFS Response Messages (continued)

Command	DFS Response Message
/DISPLAY MASTER	
/DISPLAY MODIFY	163 182 237 3431 3443 3451 3452 3455 3460 3461
/DISPLAY MSNAME	114 216 2038
/DISPLAY NODE	182 216 2103 3106 3114 3630 3639 3831
/DISPLAY OASN SUBSYS	147 182 216 3601
/DISPLAY OLDS	182 216 2038 3828
/DISPLAY PROGRAM	117
/DISPLAY PSB	147 182
/DISPLAY Q	127 147 156 163 170 216 2038
/DISPLAY RTCODE	147
/DISPLAY SHUTDOWN STATUS	134 198
/DISPLAY SUBSYS	147 182 216 3601
/DISPLAY SYSID TRAN	147
/DISPLAY TRANSACTION	146 216
/DISPLAY USER	3108 3653
/END	113 115 150 152 158 181 182 189 285 2103 2104 3106 3108 3635 3654
/EXCLUSIVE	113 115 150 152 158 181 182 189 241 2103 2104 3106 3108 3635 3654
/EXIT	107 180 181 182 183 184 189 576 577 2103 2104 2105 3106 3108 3635 3655
/FORMAT	070 182 244 3824
/IDLE	107 127 130 134 147 163 181 216 2105 2109 3633
/LOCK	062 107 114 116 117 127 146 158 162 163 182 2038 3250
/LOG	
/LOOPTEST	113 115 143 150 152 181 195 196 203
/MONITOR	099 107 113 115 130 140 150 163 182 216
/MSASSIGN	107 127 147 163 164 182 2038 2151 2152 2153 2154 2155 2156 2157 2233 2240 2244 2252 2259 2260 3200 3214
/PSTOP	099 107 113 114 115 117 130 140 146 147 150 156 163 181 182 216 232 2152 2270 2272 2273 2297 3204 3630 3633 3824
/PURGE	099 107 108 113 114 115 117 127 130 140 146 150 156 163 182 216 1953 2038 3630 3633 3824
/QUIESCE	107 130 140 158 163 181 182 216 2103 3106 3630 3632 3633 3824
/RDISPLAY	
/RMx	181 182 2038 3322
/RSTART	099 107 113 115 130 140 147 150 163 181 182 216 2103 2152 3104 3106 3108 3204 3630 3633 3639 3824
/SECURE	107 127 163 182 1953
/SMCOPY	165 181 182
/SSR	182 696 2038 3601 3609 3610 3618

Table 123. IMS Commands and DFS Response Messages (continued)

Command	DFS Response Message
/START	099 107 108 113 114 115 117 127 130 132 140 146 150 156 158 163 170 175 176 178 179 182 216 232 696 1953 2010 2020 2021 2022 2023 2024 2025 2038 2103 2105 2475 3104 3106 3108 3110 3315 3316 3319 3601 3604 3609 3630 3633 3639 3798 3799 3805 3806 3807 3808 3809 3816 3817 3824 3829 3843
/STOP	099 107 108 113 114 115 117 127 130 132 140 146 150 156 158 163 170 175 176 178 179 182 216 232 696 1953 2010 2020 2021 2022 2023 2024 2025 2038 2103 2105 2109 2475 3104 3106 3108 3110 3315 3316 3319 3601 3604 3630 3633 3639 3805 3806 3808 3824 3825 3829
/SWITCH	107 127 140 163 696 2038 3810 3813 3814 3824
/TEST	113 115 143 150 152 158 181 182 189 196 282 283 284 2103 2104 2158 3106 3108 3635 3654
/TRACE	107 108 113 130 147 150 158 163 182 216 237 279 280 281 314 775 1953 2028 2029 2030 2031 2032 2038 2093 2103 2104 2155 2460 3106 3108 3110 3630 3633 3639 3813
/UNLOCK	062 107 114 116 117 127 146 158 162 163 182 2038 3250 3813
/VUNLOAD	107 130 140 163 182 216

Restart and Recovery Considerations

Because AOI does not use the IMS message queues in this environment, messages directed to an AO application cannot be recovered at restart.

When a recoverable command is issued by the ICMD call, log record X'02' is generated when the command completes. If a failure occurs, the command is not reprocessed at restart time. If a failure occurs after a command is completed but before the command response is returned, the command response is lost. Your AO application could issue the appropriate `/DISPLAY` command if applicable to determine if the ICMD call executed successfully (as is required with asynchronous commands).

Using the Sample AO Application (DFSAOPGM)

A sample AO application is available from the IMS library IMS.SVSOURCE, member name DFSAOPGM.

Part 6. Asynchronous Data Capture

Chapter 71. Propagating Captured Data Asynchronously

As an alternative to using the Data Capture exit routine discussed in “Chapter 2. Data Capture Exit Routine” on page 41, you can propagate captured data asynchronously. This alternative is available with the addition of a logging option on the EXIT= parameter of DBDGEN you can use the logging option to capture database changes for selected segment types and environmental information in IMS Data Capture log records. Captured information is compressed using MVS/ESA compression services to minimize the space needed to store the captured data on the IMS online data sets (OLDS). Once it is stored, the captured data is available for use, for example to propagate to a DB2 environment.

This chapter describes the format of the Data Capture log records.

Related Reading: For details on the logging option, and on using the EXIT= parameter, see *IMS/ESA Utilities Reference: System*. For information about DPROPNR, see , or visit the DPROPNR home page on the Web at:

http://www.software.ibm.com/data/dpropnr

The Data Capture log records use the X'99' log code and have the following data capture subcodes to indicate the type of record being logged:

X'04'	Changed data
X'28'	End of job (EOJ)
X'30'	SETS call
X'34'	ROLS call

All of the Data Capture log records contain a common prefix, which is described in “Data Capture Log Record Prefix” on page 460. This prefix contains the subcode mentioned above that defines the type of record being logged. End of job records and SETS and ROLS call records consist entirely of this log record prefix and contain no additional information. Changed data log records, however, contain information in addition to the prefix and can span multiple physical log records to contain all of the captured data.

Changed Data Log Record

For each DL/I call, there is one logical log record that comprises all the data requested for that call. However, because all the data might not fit in one log buffer, one logical log record can actually be composed of several physical log records. To retrieve all the data logged for a particular call, you must examine the physical log records to determine where the logical log record begins and ends.

The data captured for a DL/I call is stored as elements within the changed data log record. For each DL/I call, there are multiple data elements recorded in the changed data log record. The different types of data elements are described in the next section, “Elements of Captured Data” on page 458. For an example of the recorded data elements, and the order in which they are recorded, see “Example of Logged Data Elements” on page 459.

Elements of Captured Data

The captured data elements are logged in the DCAP_DATA field within the changed data log record. The format of data elements is broken down into the header information and the actual logged data. See “Format for Data Element Header” on page 461 for the format of the log record header.

Each data element contains a 4-byte header. The header contains a 1-byte LOGID field describing the type of data being logged in this element, a 1-byte LOG_FLAG and a 2-byte LOG_LL field. The LOG_LL field contains the length of the data that follows the header. This length is used to locate the header for the next element. The header is followed by the actual data defined in the LOGID field. The data is compressed using the MVS/ESA compression services, if such services are available. If so, the COMPR_ALGORITHM field in LOG_DCAP_DATA indicates the algorithm used to compress the data.

If FLAG_1 of the changed data log record contains X'60' or X'E0' (FIRST_RECORD and FIRST_CALL flags set), the next data element is LOG_INQY_DATA. This element is only present in the first physical record of the logical log record.

The type of data element is identified in the LOGID field of the header and can be one of the following:

LOGID X'00'
There is a CAPD block for each EXIT= logging request. For the format of the CAPD block, see “CAPD Block Format (LOGID=X’00’)” on page 462.

LOGID X'04'
The DBD version for the data base, as specified on the VERSION= parameter of the DBD macro during DBDGEN. A character string of up to 63 characters.

LOGID X'08'
The concatenated key for the segment in the CAPD. The format of the physical concatenated key is a character string of up to 3,824 (255 x 15) characters.

LOGID X'0C'
The capture data segment data block (CAPD_DATA). This block is used for captured segment data, which can be path data, segment data, or before-image data. There is a CAPD_DATA block for each segment that was captured for the call. The blocks are logged in the following order: path data, segment data, before-image data. For the format of this block, see “CAPD_DATA Format (LOGID=X’0C’)” on page 464.

LOGID X'10'
The segment data for the path or segment data deleted, inserted, or replaced (for a replace, this data would represent the after-image). The length of the segment data is a character string of up to 30,700 characters.

LOGID X'14'
The before-image of the segment data changed by the replace operation. Only the first changed byte through the last changed byte is logged. The unchanged beginning and ending portions of the segment are reconstructed from the after-image contained in segment data. The before-image data is a halfword offset value (the offset of the changed data within the segment) followed by a character string of up to 30,700 characters. Only the actual

before-image is compressed. The halfword offset value is not compressed. Within the changed data log record, there may be one before-image for every changed segment.

If all the data elements do not fit in a single log record, the next log record (with the same PST_NUMBER) contains the remaining data elements, starting where the previous record left off. Before-image data, however, is unique in that the offset (2 bytes) of the data in the segment is logged preceding the changed data. Therefore, the length of the data in DCAP_DATA (LOG_LL) is 2 bytes greater than the actual length of the data logged. The offset field is not compressed.

Reducing the Amount of Captured Data

The changed data log record can contain a large amount of data for cascade deletes, especially when path data is requested. The data is staged in an internal area prior to logging. If this internal area is not large enough to hold all of the data, the ERROR_LOG flag is set in the log record to indicate that the data is not complete, although, the IMS data is committed and the call completes normally.

To avoid large amounts of data being written to the log and possible incomplete log records, specify CASCADE,KEY,NODATA,NOPATH in the EXIT= parameter in the DBDGEN if cascade is required. This specification results in just the concatenated key being written to the log during DLET operations, which significantly reduces the amount of data that is captured during a cascade delete. For data propagation to DB2 where the primary key is derived from the concatenated key, the segment data is not required for the delete in DB2.

Example of Logged Data Elements

Changed data log records contain many data elements as a result of a call. Consider a situation where a third-level segment is replaced and path data is requested to be logged (Root segment is A, second-level segment is B, and the replaced segment is C). In this situation, the data elements are logged in the following order:

1. CAPD
2. DBD Version ID
3. Physical concatenated key
4. CAPD_DATA block for A
5. Segment data for A
6. CAPD_DATA block for B
7. Segment data for B
8. CAPD_DATA block for C
9. Segment data for C
10. CAPD_DATA block for C (before data)
11. Before-image data for C

If path data had not been requested, the CAPD_DATA blocks and segment data elements for A and B would not be logged.

End of Job (EOJ) Call Log Record

The EOJ call log record (X'28' subcode) is written when a batch DL/I (DLI or DBB) program that has written changed data log records terminates normally. The record is written to indicate that the updates have been committed, since a commit record is not written to the log when a batch job terminates.

See “End of Job Call Log Record Format” on page 465 for the format of the EOJ Call Record.

SETS and ROLS Call Log Records

The SETS (X'30' subcode) and the ROLS (X'34' subcode) call log records are written whenever an application that might cause data to be captured issues a SETS or ROLS call using a token. The log records are written to indicate that any changed data log records written after the SETS call for the token used in the ROLS call will have been aborted (backed out). The records are written even if exits are defined without a logging request.

For the format of the SETS/ROLS call record, see “SETS and ROLS Call Log Record Format” on page 465.

Format of the Data Capture Log Records

This section shows the formats for the log record prefix, the data element header, and each of the data capture log records.

Data Capture Log Record Prefix

Table 124 lists the prefix for data capture log records.

Table 124. Prefix for Data Capture Log Records

Field Name	Data Type	Field Description
LL	H	The length of the log record, including a 4-byte log sequence number added by IMS to the end of the record
ZZ	H	Always zero
LOGCODE	XL1	X'99' log record code
SUBCODE	XL1	Data capture subcode
PST_NUMBER	H	PST number
RECOVTKN	XL16	The recovery token for the unit-of-recovery, which is used to associate the commit log records or the abort log records for this unit-of-recovery
STORE_CLOCK	XL8	The CPU store clock time stamp of the time the call completed and the log record was written

Changed Data Log Record Format

Table 125 lists the record formats for data capture log records.

Table 125. Format for Changed Data Log Records

Field Name	Data Type	Field Description
FLAG_1	XL1	Flag 1: Bit definitions follow:
LAST_RECORD	X'80'	The last log record written for this call. If this bit is not on, the remaining data for this DL/I call is in the log records that follow.

Table 125. Format for Changed Data Log Records (continued)

Field Name	Data Type	Field Description
FIRST_RECORD	X'40'	The first log record written for this call. If this bit is not on, the data logged at LOG_DATA_OFFSET is the data that would not fit in the preceding record.
FIRST_CALL	X'20'	The first log record written for this unit-of-recovery, so LOG_INQY_DATA is present in this log record. (FIRST_RECORD is on.)
	X'10'	Reserved for future use.
	X'08'	Reserved for future use.
ERROR_LOG	X'04'	This log record is not complete because there was an error during the processing of the log records for this call. Data for the call might not have been logged.
DBLEWORD_SEQNUM	X'02'	The log sequence number is a double word.
STCK_AT_END	X'01'	Store clock at end of the log record.
FLAG_2	XL1	Reserved for future use.
LOG_DATA_OFFSET	H	The offset in the log record where the DCAP_DATA elements start. When FIRST_RECORD is not on (indicating a split log record), this is the offset to the continuation of the data from the previous log record.
COMPR_ALGORITHM	XL1	The MVS/ESA compression algorithm used to compress DCAP_DATA.
	XL1	Reserved for future use.
LOQ_SEQ_NUMBER	CL6	The IRLM lock sequence number used when IRLM SCOPE=GLOBAL used for block-level data sharing.
LOG_INQY_DATA		The following is logged when FIRST_CALL is on.
LOG_INQY_PSBNAME	CL8	The application PSB name.
LOG_INQY_TRANNAME	CL8	The application transaction name.
LOG_INQY_USERID	CL8	The user ID for the call.
DCAP_DATA	CL0	The data element being logged.

Format for Data Element Header

Table 126 lists the format for data element headers.

Table 126. Format for Log Record Header

Field Name	Data Type	Field Description
LOGID	XL1	The type of data being logged. X'00'- CAPD block X'04'- DBD Version ID X'08'- Physical concatenated key X'0C'- CAPD_DATA X'10'- Segment data X'14'- Before-image data
LOG_FLAGS	XL1	Flags. Bit definitions follow:
COMP_DATA	X'20'	The data is compressed using MVS/ESA compression services.
	X'10'	Reserved for future use.

Table 126. Format for Log Record Header (continued)

Field Name	Data Type	Field Description
	X'08'	Reserved for future use.
	X'04'	Reserved for future use.
	X'02'	Reserved for future use.
	X'01'	Reserved for future use.
LOG_LL	H	Length of LOG_DATA.
LOG_DATA	CL0	Log data that is compressed if the MVS/ESA services are available. The 4-byte data element header is followed by the type of data that is being logged: **X'00'** CAPD block (see “CAPD Block Format (LOGID=X’00’)”). **X'04'** The log data is the DBD Version ID. **X'08'** The log data is the physical concatenated key. **X'0C'** CAPD_DATA (see “CAPD_DATA Format (LOGID=X’0C’)” on page 464). **X'10'** The log data is the Segment data. **X'14'** The log data is the Before-image data. Before-image data is preceded by a halfword offset of the changed data within the segment. The halfword offset is not compressed.

Table 127 lists the CAPD block format.

CAPD Block Format (LOGID=X’00’)

Table 127 lists the CAPD block format.

Table 127. Format for CAPD Block (LOGID = X’00’)

Field Name	Data Type	Field Description
NEXT_PTR	AL4	Internal use only. Not valid.
PREVIOUS_PTR	AL4	Internal use only. Not valid.
USER_EXIT_NAME	CL8	The name of the exit routine that is to be given control when an exit routine is requested in addition to logging. Blanks if the exit routine is not defined or is logged by DBCTL.
DATA_BASE_NAME	CL8	The name of the physical database.
DBD_VERSION_PTR	AL4	Internal use only. Not valid.
SEGMENT_NAME	CL8	The name of the physical segment that was updated and for which the log data was requested.
SEGMENT_CODE	XL1	Segment code for compare.
FLAGS	XL1	Flag: Bit definition follows:
DEDB_DB	X'80'	DEDB database.
KEY_NEEDED	X'40'	Concatenated key needed.
DATA_NEEDED	X'20'	Segment data needed.

Table 127. Format for CAPD Block (LOGID = X'00') (continued)

Field Name	Data Type	Field Description
PATH_NEEDED	X'10'	Path data needed.
	X'08'	Reserved for IMS.
	X'04'	Reserved for IMS.
	X'02'	Reserved for future use.
	X'01'	Reserved for future use.
	XL1	Reserved for IMS.
CALL_SEGMENT_LEVEL	XL1	Reserved for future use.
CALL_FUNCTION	CL4	Call function of request: **REPL:** Replace call. **ISRT:** Insert call. **DLET:** Delete call. **CASC:** Cascade delete as result of application delete call. **DLLP:** Delete of a logical parent segment through its logical path because: • It was marked as previously deleted from its physical path. • It is vulnerable to delete from both the physical and logical paths. • And, the last logical child segment is being deleted.
PHYS_FUNCTION	CL4	Physical function performed by DL/I: **REPL:** Segment physically replaced. **ISRT:** Segment physically inserted. **DLET:** Segment physically deleted. **DLPP:** Delete this segment on its physical path, but do not physically remove because logical paths to the segment from a logical child still exist.
CALL_TIMESTAMP	XL8	STCK time stamp of call completion.
AREA_NAME	CL8	The AREA name for a DEDB database.
LOWEST_LVL_KEY_OR_DATA	XL1	The lowest level number for which path data or a part of the concatenated key is added.
	XL63	Reserved for future use.
SAVED_DLTWA	AL4	Saved DLTWA for compare.
PHYSICAL_ROOT_RBA	AL4	RBA of root segment.
STORAGE_SIZE	F	Internal use only. Not valid.
CONC_KEY_PTR	AL4	Internal use only. Not valid.
CONC_KEY_LENGTH	H	Length of physical concatenated key.
ROOT_KEY_LENGTH	XL1	Length of root key.

Table 127. Format for CAPD Block (LOGID = X'00') (continued)

Field Name	Data Type	Field Description
CAPD_DATA_DIMEN	XL1	Dimension of data array. This value reflects the number of CAPD_DATA Block elements that will be logged for this call. If path data is requested, the first CAPD_DATA block element will be for the root segment, followed by CAPD_DATA block elements for each segment in the path. The next CAPD_DATA block element (first if no path data) is for the segment associated with this request, followed by the CAPD_DATA block element (if any) associated with the before-image data (physical replace operations).
CAPD_DATA_PTR(16)	16AL4	Internal use only. Not valid.

CAPD_DATA Format (LOGID=X'0C')

Table 128 lists the format for CAPD_DATA.

Table 128. Format for CAPD_DATA (LOGID = X'0C')

Field Name	Data Type	Field Description
NEXT_PTR	AL4	Internal use only. Not valid.
	AL4	Reserved for IMS.
SEGMENT_NAME	CL8	The name of the physical segment that is captured.
	XL4	Reserved for future use.
DATA_TYPE	XL1	The type of data being captured: **X'00'** Segment **X'01'** Before data **X'02'** Cascade data **X'03'** Segment path data
DATA_FLAGS	XL1	Flag byte: Bit definitions follow:
DATA_USER_IO	X'80'	Data in user I/O area.
DEL_ON_PHY_PATH	X'40'	Segment is deleted on physical path.
	X'20'	Reserved for future use.
	X'10'	Reserved for future use.
	X'08'	Reserved for future use.
	X'04'	Reserved for future use.
	X'02'	Reserved for future use.
	X'01'	Reserved for future use.
LP_KEY_LENGTH	H	The length of the segment's logical parent key concatenated in front of the segment data. Zero if segment is not a logical child.
KEY_LENGTH	H	The length of the segment's key. Zero if the segment does not have a key.
KEY_OFFSET	H	The offset of the segment's key. Zero if the segment does not have a key.
SEGMENT_LENGTH	H	The length of the segment data.
SEGMENT_PTR	AL4	Internal use only. Not valid.

End of Job Call Log Record Format

Table 129 lists the end of job call log record formats.

Table 129. Format for EOJ Call

Field Name	Data Type	Field Description
LL	H	The length of the log record, including a 4-byte log sequence number added by IMS to the end of the record.
ZZ	XL2	Always zero.
LOGCODE	XL1	X'99' log record code.
SUBCODE	XL1	X'28' log subcode.
PST_NUMBER	H	PST number.
RECOVTKN	XL16	The recovery token for the unit-of-recovery, which is used to associate the commit log records or the abort log records for this unit-of-recovery.
STORE_CLOCK	XL8	The CPU store clock time stamp of the time the program terminated and wrote the log record.

SETS and ROLS Call Log Record Format

Table 130 lists the format for the SETS and ROLS call log records.

Table 130. Format for SETS and ROLS Call

Field Name	Data Type	Field Description
LL	H	The length of the log record, including a 4-byte log sequence number added by IMS to the end of the record.
ZZ	XL2	Always zero.
LOGCODE	XL1	X'99' log record code.
SUBCODE	XL1	X'30' log subcode for SETS call; X'34' log subcode for ROLS call.
PST_NUMBER	H	PST number.
RECOVTKN	XL16	The recovery token for the unit-of-recovery, which is used to associate the commit log records or the abort log records for this unit-of-recovery.
STORE_CLOCK	XL8	The CPU store clock time stamp of the time the call completed and the log record written.
TOKEN	CL4	The SETS/ROLS token used by the call.

Part 7. DRA Exit Routines

Chapter 72. CCTL Exit Routines

There are four exit routines to which the DRA passes control. The CCTL can provide the code for any or all of these routines. If the CCTL passes an address (in the INIT request) of zero for a particular routine, the DRA uses a default exit routine.

Routine Attributes

All CCTL exit routines called by the DRA have control passed to them in 31-bit addressing mode and must return to the DRA in the same mode. Since much of the DRA has RMODE=31, registers 13 and 14 can point to locations above the 16-megabyte line. When the DRA calls the Control exit routine, the PAPL that it passes can also be above the line.

Upon entry to a CCTL exit routine, the PAPLTTOK and PAPLUSER fields are the same as they were when DFSPRRC0 first received the PAPL. (For more information on the requests use these fields, see Appendix A.) The CCTL uses the PAPLUSER field to pass information to the exit routines (for example, the address of the control blocks).

If you want the DRA to use the default exit routines supplied with IMS DB, pass a value of binary zero as the address of the exit routine in the INIT request.

Related Reading: For more information, see the section on CCTL DRA function requests in *IMS/ESA Application Programming: Database Manager*.

The functions performed by the IMS DB-supplied routines are as follows:

Routine	**Function**
Suspend	MVS wait
Resume	MVS post
Control	Always return with PAPL return code = 0 (PAPLRETC field)
Status	Free the storage area that contains user PCBs and the PCB list

To use the default Suspend exit routine and Resume exit routine, each DRA request must have the field PAPLTECB set with the address of a CCTL ECB is waited or posted.

The following sections contain the information needed for the CCTL-provided exit routines.

Related Reading: For more details on the return codes, see *IMS/ESA Messages and Codes*.

Suspend Exit Routine

The suspend exit routine receives control whenever the DRA Router routine needs to suspend a DRA request and allows the CCTL to use its own processing technique to suspend its thread. The suspend exit routine can execute before or after the Resume exit routine.

The suspend exit routine executes in the CCTL's environment. The contents of the registers upon entry are:

Register	Contents
1	Address of the PAPL
14	Return address
15	Entry point address

This routine can use a PAPL 16-word save area (PAPLSREG) to save the DRA's registers The DRA does not expect any output from this routine.

Resume Exit Routine

The resume exit routine allows the CCTL to use its own processing technique to resume a DRA request suspended by the Suspend exit routine. This routine can execute before or after the Suspend exit routine executes.

This routine receives control whenever a request has completed its process. The contents of the registers upon entry are:

Register	Contents
1	Address of the PAPL
13	Address of an 18-word save area that this routine can use to save DRA registers
14	Return address
15	Entry point address

The DRA does not expect any output from this routine.

Control Exit Routine

The Control exit routine allows the DRA to notify the CCTL about events occurring within the DRA or IMS DB. It also allows the CCTL to notify the DRA how to respond to those events. This routine receives control whenever the DRA must notify the CCTL of the following events:

- The DRA successfully identifies itself to IMS DB.
- The identify attempt to IMS DB fails.
- The CCTL's INIT request is canceled.
- The DRA fails.
- IMS DB fails.
- IMS DB terminates normally using the /CHECKPOINT FREEZE command.
- The DRA terminates due to a Control exit routine request.

The Control exit routine uses a PAPL that belongs to the DRA, never a CCTL PAPL that is a DRA request.

For all of the above events (except the last one), the CCTL must tell the DRA what action to execute next. This is done using a return code that the CCTL places in the PAPLRETC field prior to passing the PAPL back to the DRA. The DRA then acts accordingly.

The contents of the registers upon entry are:

Register	**Contents**
1	Address of the PAPL
13	Address of standard 18-word save area that the Resume exit routine can use
14	Return address
15	Entry point address

A list of possible events about which the DRA notifies the CCTL follows. With each event, the contents of the PAPL are listed with possible actions for the CCTL to take.

- After the DRA successfully identifies to IMS DB, the contents of the PAPL passed to the CCTL are:

Field	**Contents**
PAPLFUNC	Resync function code, PAPLRSYN
PAPLRSLT	Resync list address, list of recovery tokens of in-doubt UORs. First 4 bytes in the list is the number of tokens in the list. Following this number are the actual tokens, each being 16 bytes.
PAPLUSER	User data (passed on the INIT request).
PAPLDBCT	IMS DB identifier.
PAPLMTCB	Minimum thread count specified in the startup table or INIT request.
PAPLJOBN	IMS DB jobname.
PAPLCRC	IMS DB command recognition character.
PAPLIDTK	IMS DB identify token (unique store clock value representing the time the CCTL identified with IMS DB).
PAPLDSID	IMS DB ASID.
PAPLRSEN	DBRSE (IMS DB warmstandby name, =DBRSENM, IMS DB execution parameter). See *IMS/ESA Installation Volume 2: System Definition and Tailoring* for more information.
PAPLRGTY	IMS region type. The possible region types are:

PAPLDBCX	DB/DC with XRF.
PAPLDBCO	DB/DC only.
PAPLDBCL	IMS DB

After the routine has completed analyzing the PAPL, it can insert the following return codes in the PAPLRETC field to notify the DRA of the next action to take:

Code Returned	**Meaning**
0	IMS DB environment OK.
4	Terminate the DRA (the Control exit routine is not called again during this DRA session).

- After the identify to IMS DB fails, the contents of the PAPL passed to the CCTL are:

Field	**Contents**

PAPLFUNC	Failure function code
PAPLSFNC	Identify request failed subfunction code
PAPLUSER	User data (passed on the INIT request)
PAPLDBCT	IMS DB identifier
PAPLRETC	Code returned from subsystem interface or IMS DB
PAPLRCOD	Reason code. The possible reason codes are:

PAPLNTUP	Subsystem exists but is not up
PAPLNOSS	Subsystem does not exist
PAPLINT	IMS DB is in initialization process
PAPLRSTN	IMS DB waiting for restart command
PAPLRST	In restart process
PAPLBRST	DB/DC XRF backup in tracking mode
PAPLTKOV	Backup in takeover mode

After the routine analyzes the PAPL, it can insert the following data in the output fields in the PAPL to notify the DRA of the next action to take:

Field	**Contents**
PAPLDBCN	New IMS DB identifier
PAPLRETC	Code returned from the CCTL to the DRA. PAPLRETC is passed to the Control exit routine and must be reset.

Code Returned	**Meaning**
0	Issue a DFS0690 message and try to identify IMS DB again.
4	Proceed with DRA termination (the Control exit routine will not be called again).
8	Reidentify with new IMS DB identifier (in the PAPLDBCN field).

- After the DRA INIT request is canceled by a `cancel` response to the DRF690 message, the contents of the PAPL passed to the CCTL are:

Field	**Contents**
PAPLFUNC	Failure function code
PAPLSFNC	Cancelled INIT request subfunction code
PAPLUSER	User data (from the INIT request).
PAPLDBCT	IMS DB identifier.
PAPLRETC	Code returned from IMS DB.
PAPLRCOD	Reason code. The possible reason codes are:

PAPLDBNZ	IMS DB rejected identify request.
PAPLOPC	Operator responded `cancel` to DFS690 message.

After the routine has completed analyzing the PAPL, it can insert the following return codes in the PAPLRETC field to tell the DRA what to do next:

Code Returned	**Meaning**
0	Wait for a DRA TERM request.
4	Proceed with DRA termination (the Control exit routine will not be called again).

PAPLRETC is passed to the Control exit routine and must be reset.

- When the DRA fails, the contents of the PAPL passed to the CCTL are:

Field	**Contents**
PAPLFUNC	Failure function code
PAPLDRAF	DRA failure subfunction code.
PAPLUSER	User data.
PAPLDBCT	IMS DB identifier.
PAPLRCOD	Reason code

The reason codes possible are:

PAPLGMF	GETMAIN failed.
PAPLSSF	Subsystem interface failure.
PAPLDRAA	DRA abend.
PAPLESTF	Unable to establish DRA ESTAE.

The DRA expects no return code in PAPLRETC. The DRA fails and the Control exit routine is not called when the failure occurs while processing a TERM request. In this case, the PAPL return code of the returned TERM PAPL contains the failure code.

- When IMS DB fails, the DRA first issues a U002 abend to all DRA thread TCBs. In some cases, the DRA itself can also get a U002 abend and call the Control exit routine as in the previous failure event. Otherwise, the contents of the PAPL passed to the CCTL are:

Field	**Contents**
PAPLFUNC	Failure function code.
PAPLDBCF	IMS DB failure subfunction code.
PAPLUSER	User data.
PAPLDBCT	IMS DB identifier.
PAPLRETC	Code returned from IMS DB.
PAPLRCOD	Reason code. The reason code is:

PAPLABND	IMS DB abend.

The DRA expects no return code in PAPLRETC.

After the exit routine analyzes the PAPL, it can insert the following identifier and return codes in the output fields of the PAPL to notify the DRA of the next action to take:

Field	**Contents**
PAPLDBCN	New IMS DB identifier.

PAPLRETC	Code returned.

PAPLRETC is passed to the Control exit routine and must be reset.

Code Returned	Meaning
0	Wait for a DRA TERM request.
4	Wait for DRA termination.
8	Try to identify again with the new IMS DB identifier in the PAPLDBCN field.

- After IMS DB terminates using a /CHECKPOINT FREEZE command, the contents of the PAPL passed to the CCTL are:

Field	Contents
PAPLFUNC	Failure function code.
PAPLDBCC	IMS DB /CHE FREEZE subfunction code.
PAPLUSER	User data.
PAPLDBCT	IMS DB identifier.

After the exit routine analyzes the PAPL, it can insert the following identifier and return codes in the output fields of the PAPL to notify the DRA of the next action to take:

Field	Contents
PAPLDBCN	IMS DB identifier.
PAPLRETC	Code returned.

Code Returned	Meaning
0	Allow the DRA to shut itself down.
4	Terminate DRA immediately.
8	The current DRA threads are allowed to complete all current calls and are then terminated. The DRA then reidentifies with the new IMS DB identifier.

After the CCTL sets the return code equal to 0, the DRA follows the rules of the `/CHECK FREEZE` command (for example, it allows the current threads to complete their units of work). After the last thread completes, the DRA terminates. The invocation of the Control exit routine signals the completion of the DRA shutdown process.

- After the DRA terminates due to a Control exit routine request, the contents of the PAPL passed to the CCTL are:

Field	Contents
PAPLFUNC	Failure function code.
PAPLDRAF	DRA failure subfunction code.
PAPLUSER	User data.
PAPLDBCT	IMS DB identifier.
PAPLRCOD	Reason code.

The possible reason codes are:

PAPLITCF	DRA terminated due to a Control exit routine request.
PAPLMXN2	Statistic #1 (see *IMS/ESA Application Programming: Database Manager*.)
PAPLMIN2	Statistic #2 (see *IMS/ESA Application Programming: Database Manager*.)
PAPLHIT2	Statistic #3 (see *IMS/ESA Application Programming: Database Manager*.)
PAPLTIM2	Statistic #4 (see *IMS/ESA Application Programming: Database Manager*.)

Since the DRA terminated, the CCTL does not pass any return codes to IMS DB.

Control is passed to this exit routine at the end of the DRA cleanup when the DRA termination is due to a previous Control exit routine request. For example, after being notified of a IMS DB failure or a `/CHE FREEZE` command, the Control exit routine terminates the DRA.

Status Exit Routine

The function of the Status exit routine is to prevent an MVS S0C4 abend from occurring when a CCTL thread attempts to access nonexistent storage.

The DRA passes control to the Status exit routine when a TCB for a DRA thread in a scheduled state is collapsing.

Definition: The scheduled state is the time between the DRA's successful processing of a schedule request and the DRA's successful processing of one of the following thread function requests:

ABTTERM	Abort Unit of Work.
COMTERM	Commit Unit of Work.
TERMTHRD	Terminate Thread.

Related Reading: Refer to the section on CCTL DRA function requests in *IMS/ESA Application Programming: Database Manager*for a description of the thread functions.

The status exit is called to:
- Notify CCTL that the DRA thread is about to terminate for a reason other than a request from the CCTL.
- Allow CCTL to stop reference, by the CCTL thread, to storage that IMS DB acquired on behalf of the thread.
- Notify CCTL to free the storage that IMS DB acquired for the thread.

When a DRA thread successfully processes a schedule request, the address of the storage that IMS DB acquired in the CCTL's private storage is returned to the CCTL. The storage is acquired and initialized with the user's PCBLIST and PCBs. The CCTL thread uses the PCBLIST and PCBs to make DL/I requests and to receive the results of the requests. The storage is referred to as user private storage (UPSTOR).

Related Reading: Refer to the section on CCTL DRA function requests in *IMS/ESA Application Programming: Database Manager*for PAPL fields returned to CCTL when the schedule request is completed.

The CCTL thread has access to UPSTOR for the duration of the thread's scheduled state. When the scheduled state terminates normally by a request from the CCTL, IMS DB manages UPSTOR storage.

Reference to UPSTOR by the CCTL thread after the normal end of a scheduled state can result in an MVS S0C4 abend if IMS DB has freed the storage. If IMS DB allocated the same storage to another thread, reference to UPSTOR can overlay the second thread's data.

When the thread terminates abnormally during the scheduled state, the Status exit routine notifies the CCTL. The CCTL is responsible for freeing UPSTOR. The responsibility for freeing UPSTOR is assigned to the CCTL to ensure that UPSTOR is freed at the proper time.

The UPSTOR area is acquired using the GETMAIN macro by DRA thread TCBs out of subpool 0 (subpool 132 if the CCTL application is running with the public key option set).

The default Status exit routine provided by the DRA frees UPSTOR. If the CCTL chooses the default exit routine, it can incur a program check abend trying to access that storage because the CCTL might execute after the DRA has freed the storage.

The contents of the registers upon entry are:

Register	**Contents**
1	Address of the PAPL.
13	Address of standard 18-word save area that the Resume exit routine can use.
14	Return address.
15	Entry point address.

If DRA thread termination occurs during processing of a CCTL request, the CCTL's PAPL is passed to the Status exit routine. Otherwise, the DRA builds a PAPL.

The contents of the PAPL that are significant for the call are:

Field	**Contents**
PAPLUSR3	The value CCTL passed in PAPLUSR3 on the INIT request.
PAPLTOKT	The thread token set up by the CCTL. This is the token which the CCTL passed, in PAPLTTOK, on the SCHED request.
PAPLUPSA	Address of UPSTOR.
PAPLUPSL	Length of UPSTOR.

The DRA expects no return code in PAPLRETC.

Example of the DRA and Exit Routines

Table 131 illustrates how the DRA works in situations involving the functions presented in this chapter.

In this example, events in a multithreading system are shown in chronological order. To illustrate the concept of concurrent processing, the figure is split into two columns. Events that occur under the CCTL TCB (TCB with DFSPRRC0) are shown in the left column. Those events associated with the DRA or DRA thread TCBs are shown in right column. There are two CCTL threads and two DRA threads in this example.

Highlights of this example are as follows:

- One CCTL thread makes a DL/I request that fails, resulting in the Status exit routine being called.
- Another CCTL thread issues a PREP request successfully, putting that UOR in an in-doubt state in IMS DB. At this point, IMS DB fails.
- Because of the IMS DB failure, the Control exit routine is called.
- After IMS DB is restarted, the DRA identifies itself to IMS DB successfully and the Control exit routine is called for this event.
- The CCTL issues a RESYNC request to resolve the in-doubt UOR.

At the end of Table 131 are two DRA threads that were created as the result of successful SCHED requests. Assume that application programs have made several successful calls to update databases (DL/I requests). These requests, as all DRA and thread requests, are handled exactly the same with respect to multithreading as illustrated in the table found in the section on CCTL multithread examples in *IMS/ESA Application Programming: Database Manager*.

Table 131. Continuation of the Example of Events in a Multithreading System

CCTL TCB Events	DRA TCB Events
CCTL thread1 events:	
• DFSRTNA builds the DL/I PAPL and calls DFSPRRC0. • DFSPRRC0 finds the correct DRA thread. • DFSPRRC0 activates thread TCB1. • DFSPRRC0 calls the Suspend exit routine. • The Suspend exit routine suspends CCTL thread1.	
Application program2 completes. The CCTL will make sync-point requests to IMS DB to commit the processing of this UOR. The CCTL flags the UOR for application program2 as in–doubt prior to issuing a phase 1 request. The CCTL will keep a record of this in–doubt UOR until the CCTL can make a successful phase 2 call to IMS DB.	DRA thread TCB1 events: • The DRA process es the DL/I PAPL and asks IMS DB to perform a DL/I process. • The DL/I call is in progress.
CCTL thread2 events:	
• DFSRTNA issues a PREP request and calls DFSPRRC0.	

Table 131. Continuation of the Example of Events in a Multithreading System (continued)

CCTL TCB Events	DRA TCB Events
• DFSPRRC0 activates thread TCB2.	
• DFSPRRC0 calls the Suspend exit routine.	
• The Suspend exit routine suspends CCTL thread2.	DRA thread TCB2 events:
	• The DRA sends the PREP request to IMS DB. • IMS DB logs Phase 1 complete. This IMS DB UOR is now in–doubt. • The PREP request completes successfully and calls the Resume exit routine.
• The PREP request results are returned to DFSRTNA (via the PREP PAPL), and thread2 becomes inactive.	
	DRA thread TCB1 events: • IMS DB detects a DL/I failure. This failure results in termination of this DRA thread and a termination of this thread TCB. • Since this thread was in a schedule state, the DRA calls the Status exit routine and passes the DL/I PAPL back to the CCTL after putting UPSTOR information in it. • The Status exit routine associates UPSTOR with a CCTL thread, and the routine passes the PAPL to the DRA. • The DRA calls the Resume exit routine. • The DRA takes an SDUMP (see section 5) and terminates thread TCB1.
CCTL thread1 events: • After thread1 has been resumed, control is passed back to DFSRTNA. • The CCTL notices that the DL/I request failed (returning PAPLRETC=4) and takes a diagnostic dump that includes UPSTOR. • The CCTL terminates this CCTL thread and frees UPSTOR because thread1 failed. Before CCTL sends a commit request for thread2, IMS DB fails.	
	DRA TCB events:
	• The DRA calls the Control exit routine to notify the CCTL that IMS DB failed.
	• The Control exit routine returns a code (PAPLRETC=8) that tells the DRA to reconnect to IMS DB.

Table 131. Continuation of the Example of Events in a Multithreading System (continued)

CCTL TCB Events	DRA TCB Events
	• The DRA terminates any existing thread TCBs. If the CCTL makes any subsequent requests to these terminated DRA threads, the DRA will respond with a return code indicating that the request cannot be processed.
	• After IMS DB has been restarted, the DRA success fully connects to IMS DB (for more information, refer to ″INIT Request″ in Appendix A of the *Application Programming: Database Manager*).
	• The DRA calls the Control exit routine to notify the CCTL that it successfully connected to IMS DB.
Because there is an entry in the resynchronization list for the IMS DB in–doubt UOR, the CCTL creates a new task to resolve this in-doubt status.	• The DRA passes the address of the resyncronization list (PAPLRST) to the CCTL. The list contains one entry for the IMS DB in-doubt UOR for CCTL thread2.
	• The Control exit routine returns a code (PAPLRETC=0) that tells the DRA to continue running.
CCTL thread3 events:	• The DRA completes the setup process by creating new DRA thread TCBs.
• The CCTL matches the in–doubt UOR in the re synchronization list with an in.doubt UOR in its own list. The CCTL in–doubt UOR is flagged for commit processing as its Phase 2 action. • DFSRTNA issues a RESYNC request to DFSPRRC0 asking for commit processing. RESYNC is a DRA request, not a thread request. • DFSPRRC0 activates the DRA TCB to process the request and calls the Suspend exit routine.	
	• DRA calls IMS DB to commit its UOR. • After successful processing, DRA calls the Resume exit routine.
• After thread3 has been resumed, DFSRTNA receives a return code of PAPLRETC=0. • The CCTL discards its in-doubt UOR because the RESYNC request was successful.	

Part 8. External Subsystem Attach Facility

This section documents General-Use Programming Interface and Associated Guidance Information. See the Notices page to understand this classification of information.

Chapter 73. Overview

The IMS External Subsystem Attach Facility provides the capability for IMS application programs executing in IMS dependent regions to access data resources that are managed by other (external) MVS subsystems. The data access capability of IMS application programs is therefore expanded to include resources external to the IMS system in addition to DL/I databases.

An external subsystem product that wants to enable access to its data resources from IMS applications must provide functions necessary for it to "attach" to the IMS subsystem and to, jointly with IMS, coordinate data access. Part 8. External Subsystem Attach Facility describes the IMS Attach Facility programming interface presented to the external subsystem product. Installation requirements are also described (or references to other IMS publications are given where applicable).

Multiple external subsystems can only be attached by an online IMS system. These subsystems can be of the same, or of different, product types. The installation defines the external subsystems to IMS. A given IMS dependent region can have access to all external subsystems defined to the IMS system, to just a subset, or to none according to installation specifications. An application program executing in a dependent region can access more than one different subsystem. The installation defines a unique token for each subsystem, which IMS uses in routing application calls for external resources. Application program access to more than one subsystem of the same type is supported by IMS, but might not be supported by the external subsystem.

The Attach Facility provides for synchronization of external subsystem data resources with IMS data resources. For synchronization processing, IMS is the recovery coordinator and is responsible for directing commit or abort actions on behalf of its application programs. External subsystems are participants in the process and commit or abort data updates by IMS applications according to direction given by IMS. When resources are to be committed, IMS polls the participants as to whether or not they are ready to commit before giving final commit (or abort) direction.

In IMS batch, IMS is only allowed to attach to one External SubSystem (ESS). IMS expects this ESS to be the Recovery Coordinator. This ESS has no way of coordinating with any other ESS that IMS attaches to, so IMS is restricted to only one ESS attachment in batch.

The Attach Facility is different from the coordinator controller (CCTL) associated with DBCTL.

What the External Subsystem Must Provide

The External Subsystem must provide an:

External Subsystem Attachment Package (ESAP)

External Subsystem Module Table (ESMT)

Resource Translation Table (RTT)

External Subsystem Attachment Package (ESAP)

The IMS Attach Facility uses an exit routine interface. That is, to accomplish external subsystem access from dependent regions, IMS activates exit routines at certain processing points. These exit routines must be supplied by the external subsystem. The exit routine functions are prescribed by IMS; the external subsystem supplies its unique implementation. The exit routines must, in fact, provide the actual linkage to the external subsystem. IMS is not sensitive to the linkage mechanism used.

IMS loads external subsystem-supplied exit routine modules in the control region and in each dependent region that can access the external subsystem. The external subsystem can supply additional modules needed for attach exit routine processing; IMS loads these modules as well. The external subsystem modules provided for attach processing in the IMS regions make up what IMS calls the External Subsystem Attachment Package (ESAP).

External Subsystem Module Table (ESMT)

The external subsystem must specify the modules that IMS is to load in an external subsystem module table (ESMT). The external subsystem creates the module table using macros provided by IMS and makes it available to the installation. The installation specifies the name of the ESMT to IMS by including it on the definition of the external subsystem to IMS.

Related Reading: For an explanation of how to generate the ESMT, refer to "Chapter 74. Creating the External Subsystem Module Table" on page 493.

Resource Translation Table (RTT)

IMS uses a PSB (program specification block) to define the DL/I resources required by an application program. For an MPP, the PSB name is the same as the application program name; for a BMP or IFP, the PSB name can be different. The external subsystem can use a name other than the PSB name or the IMS application program name for the entity it uses to define the external subsystem resources required by the application program. If the external subsystem uses a different name, the external subsystem can provide a resource translation table (RTT) to map either PSB names or application program names to its entity names.

The external subsystem creates the RTT and makes it available to the installation. The installation specifies the name of the RTT on the definition of the external subsystem to IMS. IMS loads the RTT when it loads the ESAP.

The external subsystem is responsible for doing the actual mapping. IMS does not access the RTT; it merely loads the table and makes its address available to the ESAP. IMS does not prescribe the format of the RTT.

How External Subsystems Are Specified to IMS

The external subsystems that are to be accessed by IMS applications are defined to the IMS system in an IMS.PROCLIB member. The installation provides a PROCLIB member for the IMS control region defining all external subsystems that are to be accessed by IMS application programs running in IMS dependent regions. Included in the definition for a subsystem is the information used by IMS to direct calls from application programs to the correct external subsystem. The installation informs IMS of the definitions by specifying the control region EXEC parameter, SSM.

For each external subsystem defined to IMS, the following are specified in the PROCLIB member:

- The MVS name of the external subsystem

- The name of the external subsystem module table (ESMT) that specifies the modules in the external subsystem attachment package (ESAP)
- The language interface token (LIT) that IMS uses to route application calls to the external subsystem
- The name of a resource translation table (RTT) supplied by the external subsystem, if needed, to identify the external resources required by IMS application programs
- The command recognition character (CRC) that IMS uses to route operator commands to the external subsystem
- The region error option (REO) code indicating the action to be taken when application calls to the external subsystem cannot be processed

The installation is responsible for providing the external subsystem-supplied tables (ESMT and RTT) in the appropriate load module library.

Related Reading: The tasks the installation must perform are described in “Chapter 78. Installing External Subsystems with IMS” on page 549.

The installation has the option to supply external subsystem definitions for dependent regions. If the SSM EXEC parameter is not specified for a dependent region, the region can access all subsystems defined to the control region. If the SSM EXEC parameter is specified, the dependent region can access only those subsystems defined in the identified PROCLIB member. (The subsystems also must have been defined to the control region.) A dummy PROCLIB member (one having no definitions) is supplied if the dependent region is not to have access to any external subsystem.

The Basics of Attach Processing

An external subsystem is attached to an IMS subsystem by means of a connection established from the IMS control region to the external subsystem. A connection is also established from each dependent region that accesses the external subsystem. IMS is responsible for initiating these connections.

Subsystem Connections

The connection between an IMS application program and the external subsystem is called a *thread*. Application threads are two-way communication paths between IMS application programs and external subsystem resources. An application program can have more than one thread since it can access more than one external subsystem in one execution. However, access to multiple subsystems of the same type (multiple instances of the same subsystem type) while supported by IMS, might not be supported by the external subsystem product.

Establishing Connections

IMS uses an 'identify' process to establish a connection to the external subsystem. IMS activates an Identify exit routine contained in the ESAP to identify the control region or dependent region TCB to the external subsystem. The external subsystem can then monitor IMS TCBs in order to respond to IMS abnormal terminations. A connection is established upon successful completion of the identify process, in other words, once the region has been successfully identified to the external subsystem.

IMS provides a notify message mechanism so that if the external subsystem has not been started when IMS attempts to connect the control region, the external subsystem, once started, can notify IMS to establish the connection. If the external subsystem makes use of the notify capability, the order in which IMS and the external subsystem are started is not important.

The connection from the control region is established first before any dependent region connections are established. If the control region connection has not been established when a dependent region is started, the dependent region does not identify itself to the external subsystem. IMS uses a hierarchical relationship between control region and dependent region connections to allow the control region to act as recovery coordinator for dependent regions. If a dependent region fails, the control region takes recovery actions on its behalf.

The external subsystem can optionally provide an Initialization exit routine. IMS activates the Initialization exit routine, if provided, during control region and dependent region initialization before the region identifies itself to the external subsystem. This exit routine allows the external subsystem the chance to do any initialization processing it requires before each connection being established.

IMS can establish the control region connection automatically during control region initialization, provided the external subsystem has been started. However, the connection can be delayed to a later time. If an Initialization exit routine is not provided, or if the exit routine returns the appropriate return code, the control region identify is not done automatically. In this case, the external subsystem can activate the Subsystem Startup Service provided by IMS when it wants the connection established. Or, IMS attempts to establish the connection when a dependent region is ready to identify itself.

IMS also establishes the control region connection in response to a `/START SUBSYS` operator command.

User Authorization Processing

After a dependent region connection has been established, a signon process is performed to inform the external subsystem of the user ID associated with the IMS transaction being processed by the region. IMS activates a Signon exit routine provided by the external subsystem for this purpose. This initial signon for the region must be successful in order for a thread to be created for the application.

Signon processing can occur again during application program execution (that is, after the thread has been created). The Signon exit routine is activated for each message processed by the application program. The initial signon performed after the dependent region identify is related to the first message processed by the application (first get unique call to the message queue). Each subsequent message processed causes the Signon exit routine to be activated again to pass the new user ID. In the case of multiple mode transactions, this means that multiple signons can occur without intervening commit processing.

The external subsystem supplies a Signoff exit routine which IMS activates before terminating the dependent region connection. In the case of multiple signons for an application, signoff processing does not precede a new signon for a new message. A new signon rather replaces the previous.

Application Threads

When the application program issues its first call for data resources owned by an external subsystem, a thread is created to connect the application to the external subsystem. IMS activates a Create Thread exit routine supplied in the ESAP to identify the application program to the external subsystem. The external subsystem is expected to prepare to receive data requests from the specific application program as necessary (that is, reserve resources, create a processing structure, and so on). When the application terminates, IMS activates a Terminate Thread exit routine to terminate the thread.

Terminating Connections

A Terminate Identify exit routine must be provided in the ESAP. IMS activates this exit routine when a connection is to be terminated. Termination of the control region connection can be initiated by IMS, by the external subsystem, or by operator command (`/STOP SUBSYS`). IMS terminates the connection when it is shutting down. The `/STOP SUBSYS` command causes the connection to be terminated and also puts it in stopped status. IMS does not allow the connection to be reestablished until a `/START SUBSYS` command has been processed.

The external subsystem can request that the control region connection be terminated in one of two ways. One way is by posting a termination ECB. IMS provides, on the Identify exit routine invocation, the address of an ECB that is expected to be used by the external subsystem when it is terminating. When the external subsystem posts the termination ECB, IMS, after allowing dependent region connections to quiesce, terminates the connection and also puts the connection in stopped status (as it does for the `/STOP SUBSYS` command). The second way that the control region connection can be terminated is by activating the IMS-supplied Subsystem Termination Service from an external subsystem exit routine.

After activating the Terminate Identify exit routine in the control region, IMS activates the Subsystem Termination exit routine supplied in the ESAP. This exit routine, which can be thought of as the reverse of the Initialization exit routine, might be used by the external subsystem to reset work areas or free storage, for example.

A dependent region connection is maintained for as long as the region is active unless IMS has been requested, either by the external subsystem or by an IMS `/STOP SUBSYS` command, to terminate the (control region) connection. In general, it is only when IMS has been requested to terminate the connection that the external subsystem Terminate Identify exit routine is driven for dependent regions. Thus, the exit routine is not necessarily activated when a dependent region terminates. This is true also for the Signoff exit routine. Terminate Identify exit routine invocation always follows Signoff exit routine invocation.

The external subsystem is expected to monitor, via MVS end-of-task exit routines, the IMS TCBs identified to it and to perform the necessary signoff and terminate identify processing when an identified TCB ends.

Terminate Thread exit routine invocation always precedes normal termination of the dependent region connection if the region had a thread to the external subsystem. Thus the Terminate Thread exit routine is activated before the Signoff and Terminate Identify exit routines are activated (if they are) or before the dependent region is terminated.

Since IMS does not allow dependent region connections to exist unless the control region has a connection, the Terminate Identify exit routine is not activated for the control region until after each dependent region has either terminated or had its Terminate Identify exit routine activated.

The Subsystem Termination exit routine is not activated for dependent regions.

Inquiry Parameter Processing

The INQ parameter is only checked when the IMS transaction issues a Create Thread exit routine. The INQ parameter on subsequent transactions is not checked. Therefore, if any updates are to be done in a Fast Path region between the Create Thread exit routine and the Terminate Thread exit routine, the inquiry flag in the first transaction must be INQ=NO.

For example, if the first transaction that calls DB2 from a given Fast Path region is only going to read the DB2 data and not update it, the transaction will set the INQ=YES flag in the Create Thread Parameter list indicating that this first transaction and all subsequent transactions in that Fast Path region are treated as inquiry only transactions. If a subsequent transaction running under the same Fast Path region calls DB2 for update, the thread to that Fast Path region will still be set to INQ=YES, even though the transaction is correctly defined as INQ=NO. This will result in an SQLCODE817 error.

Application Call Processing

Once a thread from the application program to the external subsystem has been created, application calls for external data resources are passed to the Normal Call exit routine supplied in the ESAP. The language interface link-edited with the application provides the language interface token (LIT) for the external subsystem when it activates IMS to process calls to the external subsystem. The installation specifies a unique LIT for each external subsystem it defines to IMS. IMS matches the LIT provided by the language interface stub with the LIT specified in the definition to route the call to the external subsystem.

Resource Coordination

IMS, as recovery coordinator, directs commit processing for updates to external subsystem resources initiated by IMS application programs. IMS uses a two-phase commit process to synchronize resources across external subsystems. External subsystems are participants in the process. In the first phase of the commit process for an application, IMS polls the participants for a vote as to whether or not they are prepared to commit the updates. In the second phase, IMS directs the participants to commit or to abort. If all participants voted 'yes' on the first phase, IMS directs them to commit on the second phase; otherwise, IMS directs them to abort.

When an external subsystem determines that its resources are associated with non-update transactions (for which commit processing is not necessary), the external subsystem can perform all commit processing during the first phase, eliminating the need for the second phase. In this case, the external subsystem returns to IMS from the Commit Prepare exit routine with return code X'C' indicating that the first phase successfully completed and the second phase is not required. IMS will not initiate the second phase of commit processing for this external subsystem.

IMS uses a 16-byte recovery token to identify a unit of work across one or more subsystems. The recovery token for a unit of work is initially passed on the Signon exit routine invocation.

When application updates are to be committed, IMS activates the Commit Prepare exit routine supplied by the external subsystem. The associated recovery token is passed on the invocation. The external subsystem indicates, via the return code from the exit routine, whether or not it is prepared to perform commit processing for the recovery token.

For the second phase of the commit process, if it is required, IMS can activate either of three external subsystem exit routines: the Commit Continue exit routine, the Abort Continue exit routine, or the Terminate Thread exit routine. When the application is not terminating and all participants are prepared to commit, IMS drives the Commit Continue exit routine. At completion of the commit process, the application will continue processing the current PSB on the existing thread. When the updates are to be aborted but the application is not terminating, or being terminated, the Abort Continue exit routine is activated. In this case, the application will continue processing under the same recovery token. When an application is executing in a Distributed Syncpoint environment (also known as a Protected Conversation environment) and requires a subsystem SIGNON, IMS obtains the XID token and places its address in the exit parameter list before calling the subsystem's SIGNON exit.

The external subsystem Terminate Thread exit routine must be able to process the second phase of commit. At application termination, IMS passes the recovery token and a commit option on the Terminate Thread exit routine invocation. The commit option indicates whether to commit or abort outstanding updates.

When IMS, the external subsystem, or an application program terminates abnormally, units of work that have not been committed or aborted are left outstanding. To resolve outstanding units of work, IMS activates the external subsystem Resolve In-doubt exit routine. IMS always activates the Resolve In-doubt exit routine at least once after establishing the control region connection. IMS activates the exit routine once for each outstanding recovery token indicating whether to commit or abort the unit of work. When there are no units of work to be resolved or when IMS has exhausted the list of outstanding recovery tokens, IMS activates the exit routine to inform the external subsystem of that fact. When IMS encounters an outstanding recovery token associated with RRS, IMS will delay the subsystem Resolve In-Doubt exit call until RRS or the IMS user has indicated (ABORT or COMMIT) which action to take. When called for RRS Resolve In-Doubt, it is the subsystem's responsibility to ensure that revocery tokens are resolved in their proper order.

IMS maintains outstanding recovery tokens across normal (warm) and emergency restarts of IMS, and reconnections of the subsystems. IMS permits a connection without all recovery tokens being resolved (that is, the Resolve In-doubt exit routine return code can indicate that the recovery action was not taken). IMS destroys outstanding recovery tokens when it is cold started.

The Resolve In-doubt exit routine is also used to coordinate resources in the event of abnormal termination of an application program. Following an application program abend, the exit routine is activated from the control region if the application had a thread connection to the external subsystem.

External Subsystem Command Support

IMS provides a command, /SSR, which allows the IMS operator to send commands to the external subsystem. To receive commands from IMS the external subsystem must supply a Command exit routine. IMS passes the command contained in the /SSR input to this exit routine. AOI (automated operator interface) programs can also send commands to external subsystems using /SSR. The /SSR command input contains identification, a command recognition character (CRC), of the external subsystem to which the command is directed. The CRC for a subsystem is specified as part of the definition of the subsystem to IMS.

Related Reading:
- For information on the /SSR command, see *IMS/ESA Operator’s Reference*.
- For information on the CRC, see “CRC command recognition character” on page 552.

IMS Services Available to the ESAP

IMS provides exit routines that an external subsystem can activate to access certain IMS system services. The external subsystem can:
- Request that a connection be initiated (Subsystem Startup Service).
- Request that connections be quiesced (Subsystem Termination Service).
- Have a log record written to the IMS log (Log Service).
- Have a message sent to an IMS destination (Message Service).

Related Reading: For a description of these exit routines, refer to “Chapter 77. IMS System Service Exit Routines” on page 541.

Chapter 74. Creating the External Subsystem Module Table

The external subsystem creates the external subsystem module table (ESMT) to supply definitions of the external subsystem modules that IMS is to load. External subsystem exit routine modules, as well as any other modules needed in the ESAP, are defined in the ESMT. The installation provides the name of the ESMT to IMS as part of the definition of the external subsystem. During initialization for attach processing in the control and dependent regions, IMS loads the ESMT and then loads the modules defined therein.

The external subsystem optionally provides in the ESMT definitions of work areas needed for its ESAP. If work area definitions are provided, IMS obtains the specified work area storage in each region after loading the defined modules.

IMS provides two macros to be used by the external subsystem to create the ESMT. The DFSEMODL macro is used to define the ESAP modules that IMS is to load. The DFSEWAL macro is used to define the work areas that IMS is to create. A series of DFSEMODL statements defining modules, optionally followed by DFSEWAL statements defining workareas, and ending with a DFSEMODL statement specifying END=LAST, generates the table.

DFSEMODL Macro

The ESMT is generated from a series of DFSEMODL statements, one for each module definition. In addition to module definition information, information about the control block that is to contain the addresses of the modules when they are loaded is also supplied on DFSEMODL statements. IMS creates this control block before it loads the modules.

IMS provides the capability for up to three sets of modules to be loaded and anchored on separate control blocks. Accordingly, the ESMT consists of one to three subtables, each containing the specifications for a set of modules and their module address control block. The module address control block for external subsystem exit routines is the EEVT.

When the module address control block is created, IMS stores its address into a source control block, which is the EEVTP (EEVT prefix).

The format of the DFSEMODL macro is:

```
(label)   DSFEMODL DSNAME=,SOURCE=,MODNAME=,DSLABEL=,
                   SUBPOOL=,OPTION=,END=
```

where:

(label) Is optional. If coded, the macro generates ESMT subtable labels. The last label on a macro statement in the series from which a subtable is generated is used as the subtable label.

The following parameters provide control block information and need only be specified once per subtable (for example, on the first DFSEMODL statement in the series). If specified on more than one statement, the first specifications encountered are used in generating the table.

DSNAME= (p1,p2,p3)

p1 Name of the module address control block. The name must be specified (at least on one DFSEMODL statement) for each ESMT subtable.

p2 Module address control block size. The size must be specified. IMS obtains storage of the specified size to create the module address control block.

p3 Subpool number for the module address control block storage request. This parameter is optional. If 251 is not specified, IMS obtains the storage from subpool 230.

SOURCE= (p1,p2)

p1 Name of the source control block. This parameter is required. DFSEEVTP must be specified. (See the following discussion.)

p2 Label in the source control block of the location to store module address control block address. This parameter is required.

The following parameters provide module definition information.

MODNAME= Name of the module IMS is to load. MODNAME must be specified on all DFSEMODL statements that do not specify the END parameter. (END can be specified with or without MODNAME.)

DSLABEL= Label in the module address control block of the location to store the module address after it is loaded. DSLABEL must be specified (when MODNAME is specified).

SUBPOOL= Subpool into which IMS is to load the module. SUBPOOL must be specified (when MODNAME is specified). For the control region, IMS loads the module into the subpool specified. For dependent regions, IMS loads the module into subpool 251 if `SUBPOOL=251` is specified; otherwise, the module is loaded into subpool 230. Valid specifications are: 0, 229, 230, 231, 241, 251, 252.

OPTION= (p,p)

This parameter is optional. Two options, NOCTL and NODEP, are supported. (Position of an option in the subparameter list is not important.)

NOCTL
The module is not to be loaded in the control region.

NODEP
The module is not to be loaded in dependent regions.

The END parameter controls ESMT generation.

END=

YES Must be specified to indicate the end of a subtable in the ESMT being generated. END=YES is used only when the ESMT is to contain more than one subtable. It is specified to end each subtable except the last (or only). DFSEMODL statements for the next subtable in the ESMT are to follow the END=YES specification.

LAST Must be specified on the last DFSEMODL definition statement for the ESMT being generated. The next DFSEMODL or DFSEWAL statement (if any) causes a new ESMT generation to be started.

You must link-edit the ESMT module into a program library (RESLIB) using a linkage editor ENTRY statement that specifies MAINEP as the entry point. A table definition header is generated at the end of the ESMT module. The `ENTRY` statement allows IMS to correctly reference the header for subsequent processing.

DFSEMODL supports an execute form (MF=E) for internal use only. It cannot be used for ESMT generation. The list form (MF=L) described is the default.

Mapping DSECTs for the module address and source control blocks must be included in the ESMT generation source, otherwise the assembly will fail.

The following restrictions apply to external subsystems:

- Specifying the source control block.

  DFSEEVTP must be specified as the source control block name (SOURCE(p1)) for all IMS-defined subtables. Otherwise, although DFSEMODL accepts other specifications, the module load process will fail, prohibiting a connection for the region experiencing the failure. The user must use the EEVTP mapping layout as earlier defined, as this is the layout that IMS expects.
- Defining subsystem exit routine modules.

  DFSEEVT must be specified as the module address control block name (DSNAME(p1)) for the subtable that contains the subsystem exit routine module definitions.

  The module address control block size (DSNAME(p2)) must be specified according to the size indicated by the EEVT mapping (EEVTLGTH) as shown in “Control Block Mapping” on page 503.

  EEVPEEA must be specified as the label in the source control block (SOURCE(p2)) to anchor the module address control block (EEVT). IMS does not check for this (nor does the macro) but uses the offset generated from the label specified to store the address. If the offset is incorrect, IMS will not be able to activate the exit routines.

  The label (DSLABEL) specified for a particular subsystem exit routine module is used to generate the offset that IMS uses to store the exit routine address in the module address control block (for example, in the EEVT). Thus these labels must be specified according to the EEVT mapping.
- Generating additional subtables.

  The ESMT must always have one subtable containing definitions for exit routine modules. The external subsystem could choose to have other modules needed in its ESAP anchored on a separate control block, which means that another subtable would be generated.

  Although the DFSEMODL macro does not restrict the number of subtables that can be generated, problems can occur during processing if more than three (3) are generated. For each subtable, IMS creates a module address control block and stores its address in the EEVTP. There are only three fields in the EEVTP that could be used as anchors for these control blocks, one of them being the anchor for the EEVT.

  The EEVTLDIR and EEVPEWA fields are not used by IMS and thus are available for this purpose. The discussion on defining external subsystem work areas in “DFSEWAL Macro” on page 496 suggests how EEVPEWA might be used to anchor a work area address control block.

- Defining external subsystem-unique modules.

  If the ESAP needs non-IMS exit routine modules (for example, modules that the external subsystem activates without any knowledge of or support from IMS), the external subsystem can define these modules in an additional subtable as previously discussed. The external subsystem must supply the mapping DSECT for the module address control block for these modules.

  Recommendation: Although other modules can be defined in the subtable containing exit routine modules and the EEVT size extended to include their addresses, IMS recommends that this not be done. The EEVT is an IMS control block which IMS reserves the right to extend at any time, which could require the external subsystem to regenerate the ESMT and re-compile modules.

DFSEWAL Macro

The work areas that IMS is to create for the external subsystem must be defined by including DFSEWAL macro statements along with the DFSEMODL statements provided for ESMT generation. The DFSEWAL statements, one for each work area defined, follow the DFSEMODL statements, except that the last statement in the series must be a DFSEMODL statement specifying END=LAST. The DFSEWAL statements cause a table of work area definitions to be built in the generated ESMT.

Work areas can be defined in each subtable generated in the ESMT. At least one module must be defined in each subtable. If a subtable is generated containing only work area definitions, an error occurs during IMS processing of the ESMT.

IMS creates the work areas defined in a subtable after loading the modules defined in the subtable. IMS stores the addresses of the created work areas in a work area list control block. This control block is also defined via the DFSEWAL macro and can either be contained in the module address control block for the subtable or be created as a separate control block. For this discussion, EWAL is used to refer to the external subsystem work area list control block.

Recommendation: It is recommended that the EWAL be contained in the module address control block rather than created as a separate control block. The reason is that when IMS creates the EWAL, its address is not (explicitly) provided to the external subsystem. If, instead, the EWAL is contained in the module address control block, which IMS anchors in the EEVTP, the external subsystem specifies its location (via DFSEWAL) and thus knows how to access it. (When IMS creates the EWAL, it stores its address in the in-storage ESMT for internal use. The format of the ESMT is not included in the documented attach interface.) Figure 24 shows a representation of the relationship between the EWAL, EEVTP, and EEVT.

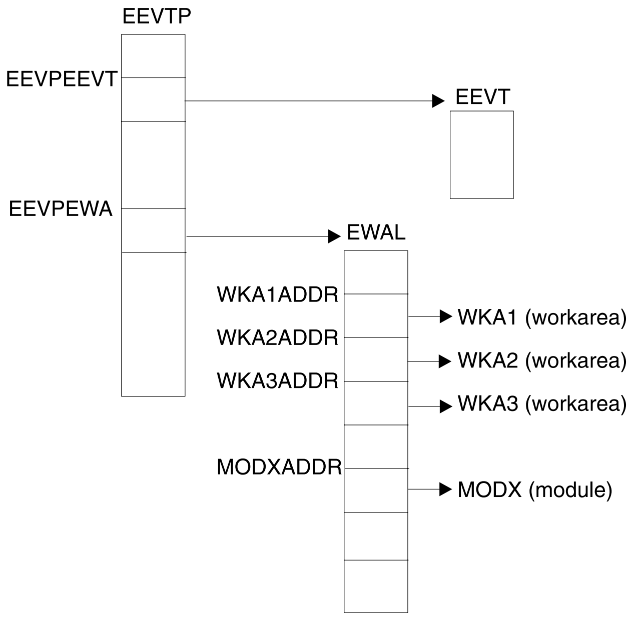


Figure 24. EWAL, EEVTP, and EEVT Relationship

If the external subsystem wants IMS to create work areas for its ESAP, it should define two (possibly three) subtables in the ESMT. Modules definitions would be contained in one subtable. The module address control block for this subtable is the EEVT. The second subtable would contain work area definitions. The module address control block for this subtable would either contain the EWAL or be used as the EWAL, and would be anchored in the EEVTP along with the EEVT. (As mentioned under “DFSEMODL Macro” on page 493, modules could be defined in two subtables: one for exit routines and one for other external subsystem modules that are activated by exit routines.)

An example is provided later in this section that illustrates how the external subsystem might specify work area definitions.

The format of the DFSEWAL macro is as follows:

```
DFSEWAL DSNAME,SOURCE=,WALSP=,NAME=,DSLABEL=,
        SUBPOOL=,LV=,OPTION=
```

where:

The following parameters provide control block information and need only be specified once per subtable (for example, on the first DFSEWAL statement in the series). If specified on more than one statement, the first specifications encountered are used in generating the table.

DSNAME= (p1,p2)

p1 Name of the work area list control block mapping DSECT. The DSECT name must be specified. If IMS creates the work area list, this name is given to the job pack entry for the storage acquired.

p2 Work area list size. If the size is specified, IMS obtains storage of the specified size to create the work area list. If the size is not specified IMS does not create the work areas unless the source control block for the work area list

(DFSEWAL SOURCE(p1) specification) is the module address control block specified for the modules defined in the subtable (DFSEMODL DSNAME(p1)). (See the following discussion.)

SOURCE= (p1,p2)

- **p1** DSECT name for the control block in which the work area list is to be anchored. This parameter must be specified. (See the following discussion).
- **p2** Label in the source control block DSECT of the location that is to contain the work area list. This parameter is required. IMS does not store the work list area address into this control block. (See the following discussion.)

WALSP= Subpool number for the work area list storage request. This parameter is optional. If `WALSP=251` is not specified, IMS obtains the storage from subpool 230.

The following parameters provide work area definition information and must be specified on each DFSEWAL statement.

NAME= The name given to the job pack directory entry created for the work area storage acquired. This parameter is required.

DSLABEL= Label in the work area list control block DSECT of the location into which IMS is to store the work area address. DSLABEL must be specified.

SUBPOOL= Subpool from which IMS is to obtain storage for the work area. The subpool must be specified. IMS acquires subpool 251 storage if `SUBPOOL=251` is specified; otherwise, the work area is created in subpool 230. The macro allows 0, 229, 230, 231, 241, 251, or 252 to be specified.

LV= Work area size. The size must be specified.

OPTION= (p,p)

This parameter is optional. Two options, NOCTL and NODEP, are supported. (Position of an option in the subparameter list is not important.)

NOCTL
Work area is not to be created in the control region.

NODEP
Work area is not to be created in dependent regions.

Mapping DSECTs for all referenced control blocks must be included in the ESMT generation source, otherwise the assembly will fail.

The source control block DSECT name and label must be specified. However, IMS does not store the EWAL address into this control block.

To indicate that the EWAL is to be contained in the module address control block:

- The size for the EWAL (DFSEWAL DSNAME(p2)) must not be specified.
- The module address control block DSECT name (DFSEMODL DSNAME(p1)) must be specified as the EWAL source control block DSECT name (DFSEWAL SOURCE(p1)).

- The source control block label (SOURCE(p2)) must specify the location of the work area list in the module address control block.

If the size for the EWAL is specified, IMS obtains storage for the EWAL without checking if the module address control block was specified as the EWAL source. If the EWAL size is not specified and the module address and EWAL source control block DSECT names do not match, IMS does not create the work areas. (IMS does not know the address of the source control block. IMS does not indicate that the work areas were not created.)

IMS reserves the EEVPEWA field in the EEVTP control block for the address of an EWAL. The following example illustrates how definitions can be specified to anchor a work area list in this field. What really happens is that a module address control block is created, anchored at EEVPEWA, and used as the EWAL.

In Figure 25:

- The existence of a DSECT named ESSEWAL created by the external subsystem to map the EWAL is assumed.
- Two subtables are defined for completeness:
  - The first subtable contains exit routine module definitions.
  - The second subtable contains work area definitions:
    - A module is defined in this subtable with EEVPEWA specified as the anchor field for the module address control block. (If the external subsystem does not really want a module loaded for this subtable, both the NOCTL and NODEP options can be specified.)
    - The module address control block DSECT, ESSEWAL, is specified as the EWAL source control block DSECT and the EWAL size is not specified, indicating that the EWAL is to be contained in the module address control block.
    - ESSEWAL is also specified as the label in the source block for the EWAL, indicating that the EWAL starts at offset zero in the module address control block. Thus, the module address control block itself is the EWAL, anchored at EEVPEWA in the EEVTP.

```
DFSEMODL DSNAME=(DFSEEVT,68,230),SOURCE=(DFSEEVTP,EEVPEEA),
            MODNAME=INITEXIT,DSLABEL=EEVTINIT,SUBPOOL=230
   DFSEMODL MODNAME=IDEXIT,DSLABEL=EEVTID,SUBPOOL=230
   DFSEMODL MODNAME=RIDEXIT,DSLABEL=EEVTRID,SUBPOOL=230,
            OPTION=NODEP
                  .       .       .
                  .       .       .
                  .       .       .
   DFSEMODL MODNAME=CMDEXIT,DSLABEL=EEVTCMD,SUBPOOL=230,
            OPTION=NODEP
   DFSEMODL END=YES
   DFSEMODL DSNAME=(ESSEWAL,40,230),SOURCE=(DFSEEVTP,EEVPEWA),
            MODNAME=MODX,LABEL=MODXADDR,SUBPOOL=230
```

Figure 25. Example (Part 1 of 2)

```
DFSEWAL  DSNAME=(ESSEWAL),SOURCE=(ESSEWAL,ESSEWAL),WALSP=230,
         NAME=WKA1,DSLABEL=WKA1ADDR,SUBPOOL=230,LV=200
DFSEWAL  NAME=WKA2,DSLABEL=WKA2ADDR,SUBPOOL=230,LV=100
DFSEWAL  NAME=WKA3,DSLABEL=WKA3ADDR,SUBPOOL=230,LV=100
DFSEMODL END=LAST
```

Figure 25. Example (Part 2 of 2)

Chapter 75. External Subsystem Exit Routines

IMS activates external subsystem-supplied exit routines to perform prescribed subsystem unique attachment functions. IMS uses the module names in the external subsystem module table (ESMT) specified for the control region to load the exit routines during control region initialization. The ESMT specified (or defaulted to) for a dependent region is used to load the exit routines into the dependent region.

As indicated in the individual exit routine descriptions later in this chapter, most of the exit routines execute functions that are required for attach processing; others are optional. When an exit routine required to support connection processing is not present, IMS terminates the connection to the external subsystem, if one exists, and issues an informational message (DFS3068I). If an application program is involved, it is terminated with a user abend (U3049).

The external subsystem exit routines are organized alphabetically and described beginning on page505.

General Exit Routine Interface

This section describes general interfaces for all the External Subsystem exit routines. You need to familiarize yourself with these interfaces.

Exit Parameter List (EPL)

IMS activates an external subsystem exit routine, passing the address of an exit parameter list (EPL) in register 1 (see Figure 26). The EPL contains the addresses of the parameters required by the exit routine. IMS passes to an exit routine only the specific parameters it requires, so the contents and length of the EPL differ between exit routines. The parameters for each exit routine are specified in the individual exit routine description sections that follow.

The general format of the EPL is an array of fullword fields (4-byte fields, fullword aligned), each containing the address of a parameter required for the exit routine being activated. The first word in the EPL always contains the address of a 4-byte parameter count field. The binary value in the count field is the number of parameters being passed minus the count parameter.

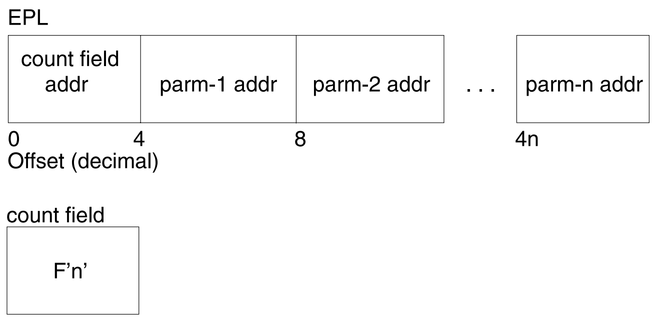


Figure 26. Exit Parameter List

Contents of Registers

Upon entry, the exit routine must save all registers using the provided save area. The registers contain the following:

Attach Facility

Register	Contents
1	Address of exit parameter list (EPL).
13	Address of save area. The exit routine must not change the backward chain field, but it can alter the forward chain field.
14	Return address to IMS.
15	Entry point of exit routine.

Before returning to IMS, the exit routine must restore all registers except for register 15, which must contain a return code. IMS provides one standard register save area (address in register 13) in the appropriate storage protect key into which the exit routine can save the entry register contents. The save area backward chain field must not be altered (such as to chain the save area into a save area set). The exit routine can alter the forward chain field.

Return Codes

Return codes are exit routine specific. For each exit routine, the supported return codes are described in the exit routine description section later in this chapter. The return codes are shown in hexadecimal format. Return Code 20 is supported for all exit routines and is described below.

Unsupported Return Codes

If register 15 on return from an exit routine contains a return code that is not supported for the exit routine, it is treated as an error. IMS terminates the connection for the region that activated the exit routine if one exists. If an application program is involved, it terminates with a U3049 abend.

Return Code 20

Return code 20 is used by all exit routines to indicate a 'should not occur' condition and is described as follows:

Return Code	Meaning
20	Should not occur. The exit routine encountered a 'should not occur' condition while processing the request. Such conditions include invalid save areas, protocol violations, invalid work areas, and invalid parameter lists. Action: • If an application program is involved, it terminates with abend U3044. If the external subsystem does not respond or responds incorrectly to the control region echo request, the connection to that subsystem terminates. • If the external subsystem does respond, the identify for the dependent region terminates. A subsequent external subsystem request causes the structure to be rebuilt. • If a connection exists when the error is encountered, it terminates by activating the Terminate Identify exit routine (see “Terminate Identify Exit Routine” on page 527).

Exit Routine Interface Control Blocks

This section describes the prefix for the external entry vector table and the vector table itself.

External Entry Vector Table Prefix (EEVTP)

The address of an **external entry vector table prefix (EEVTP)** control block is always passed in the EPL on exit routine activations. The EEVTP is the primary external subsystem interface control block and contains the:

- Address of the external entry vector table (EEVT)
- Address of the resource translation table (RTT)
- Environment indicator (control or dependent region TCB)
- Address of the IMS service exit routine router module

External Entry Vector Table (EEVT)

The external entry vector table (EEVT) contains the addresses of external subsystem exit routine modules. IMS gets exit routine addresses from this control block to activate the exit routines. IMS creates an EEVT (and EEVTP) in the control region and in each dependent region before loading the modules defined in the ESMT into the region. When the modules are loaded their addresses are stored in the EEVT.

The EEVT is an IMS control block, however, module addresses are placed in the control block based on the module definitions contained in the ESMT. Therefore, the external subsystem, in creating the ESMT, must make sure that exit routine module definitions provide for placement of exit routine addresses in the EEVT according to the EEVT mapping layout used by IMS. The ESAP can manipulate addresses in this vector table if it chooses.

Related Reading:

- “Chapter 74. Creating the External Subsystem Module Table” on page 493 describes how external subsystem modules are defined.
- The EEVT layout is shown in “Control Block Mapping”.

In addition to exit routine modules, the external subsystem can define other modules in the ESMT, for example, modules that would be activated by exit routines and not by IMS. IMS loads all modules defined in the ESMT and stores their addresses as specified in the definitions.

Because of how definition and loading of external subsystem modules is done, it is possible for the external subsystem to 'extend' the EEVT to include the addresses of nonexit routine modules.

Recommendation: It is recommended that this not be done. IMS might add fields to the EEVT at a later time, in which case, the external subsystem might have to respecify module definitions (that is, regenerate the ESMT) and recompile modules.

Related Reading: For suggestions about how to define nonexit routine modules and have their addresses stored in a separate control block, refer to “Chapter 74. Creating the External Subsystem Module Table” on page 493.

Control Block Mapping

The following DSECTs map the control blocks discussed in Exit Routine Interface Control Blocks.

DFSEEVTP

The DFSEEVTP DSECT maps the EEVTP control block. (The EEVTP is the prefix of the EEVT and contains the address of the EEVT.) The following fields are of interest to the external subsystem:

Offset Hex	Field Length (Hex)	Field Name	Description
0	4	EEVPNAME	EYECATCHER - ’EEVP’
8	4	EEVPEEA	EEVT ADDRESS
10	4	EEVPEWA	AVAILABLE FOR EXTERNAL SUBSYSTEM
14	4	EEVPRTA	RECOVERY TOKEN ADDRESS
1C	4	EEVPRTTA	RESOURCE TRANSLATION TABLE ADDRESS
20	4	EEVTLDIR	AVAILABLE FOR EXTERNAL SUBSYSTEM
28	4	EEVPESGL	DFSESGL0 ADDRESS
2E	1	EEVPF1	ENVIRONMENT INDICATORS
-	-	EEVPCR	= X'01';CONTROL REGION
-	-	EEVPMPP	= X'02';MPP DEPENDENT REGION
-	-	EEVPBMP	= X'04';BMP DEPENDENT REGION
-	-	EEVPIFPN	= X'08';FAST PATH NON-MESSAGE DRIVEN
-	-	EEVPIFPM	= X'10';FAST PATH MESSAGE DRIVEN
-	-	EEVPBMPN	= X'20';NON-MESSAGE DRIVEN BMP
-	-	EEVPBDB2	= X'80';BATCH DB2 REGION
2F	1	EEVPF2	ENVIRONMENT INDICATORS
-	-	EEVPDRPG	= X'01';RUNNING UNDER DEPENDENT REGION.. ..PROGRAM CONTROLLER TCB
34	8	EEVPSOTN	SIGNON TOKEN
3C	4	EEVPESMT	ESMT ADDRESS
40	4	EEVPSVA	EESV ADDRESS
-	-	EEVPLGTH	= X'44';LENGTH OF EEVTP

DFSEEVT

The DFSEEVT DSECT maps the EEVT control block. The following fields are of interest to the external subsystem:

Offset Hex	Field Length (Hex)	Field Name	Description
0	4	EEVTNAME	EYECATCHER - ’EEVT’
4	4	EEVTINIT	INITIALIZATION EXIT ADDRESS
8	4	EEVTID	IDENTIFY EXIT ADDRESS
C	4	EEVTRID	RESOLVE-IN-DOUBT EXIT ADDRESS
10	4	EEVTSO	SIGNON EXIT ADDRESS
14	4	EEVTCT	CREATE THREAD EXIT ADDRESS
18	4	EEVTCP	COMMIT PREPARE EXIT ADDRESS
1C	4	EEVTCC	COMMIT CONTINUE EXIT ADDRESS
20	4	EEVTA	ABORT EXIT ADDRESS
24	4	EEVTTT	TERMINATE THREAD EXIT ADDRESS
28	4	EEVTSF	SIGNOFF EXIT ADDRESS :row,

Offset Hex	Field Length (Hex)	Field Name	Description
2C	4	EEVTTI	TERMINATE IDENTIFY EXIT ADDRESS
30	4	EEVTSNO	SUBSYSTEM NOT OPERATIONAL EXIT ADDR
34	4	EEVTSST	SUBSYSTEM TERMINATION EXIT ADDRESS
38	4	EEVTNC	NORMAL CALL EXIT ADDRESS
3C	4	EEVTECHO	ECHO EXIT ADDRESS
40	4	EEVTCMD	COMMAND EXIT ADDRESS
44	4	EEVTCV	COMMIT VERIFY EXIT ADDRESS
48	4	EEVTIC	NOT USED
4C	4	EEVTABE	NOT USED
50	4		RESERVED
54	4		RESERVED
-	-	EEVTLGTH	= X'58';LENGTH OF EEVT

Abort Continue Exit Routine

The Abort Continue exit routine is activated by IMS for all transaction types. The external subsystem resource managers hold onto the resources they have acquired on behalf of the application. The application will continue using the current recovery token. The Abort Continue exit routine is activated when:

- The application issues an IMS DL/I ROLB call.
- An external subsystem votes 'no' to a commit prepare request.

Activating the Routine

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EEVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Offset Hex	Decimal	Content
0	0	Address of the parameter count field. The count field contains the value F'2'.
4	4	Address of the EEVT prefix.
8	8	Address of the 16-byte recovery token associated with this instance of the transaction. The recovery token identifies the unit of work across one or more subsystems.

Contents of Register 15 on Return

Processing of return codes received from the exit routine:

Return Code	Meaning
0	Abort Continue successful. **Action:**IMS continues normal processing.

Return Code	Meaning
4	Abort Continue unsuccessful. ESAP or external subsystem processing of the request failed. **Action:** IMS terminates the application with abend U3045 (the input message is discarded; DL/I resources are backed out). The control region performs resolve in-doubt processing for the recovery token. (See “Resolve In-Doubt Exit Routine” on page 519 for more information about this exit routine.) The dependent region is terminated; which implicitly terminates the dependent region connection to the external subsystem; the Signoff and Terminate Identify exit routines are not called). BMP jobs must be resubmitted; they resume processing at the prior commit point.
20	Should not occur. See “Return Codes” on page 502 for more information.

Command Exit Routine

IMS activates the optional external subsystem Command exit routine when IMS discovers the subsystem’s unique command recognition character (CRC) as the first non-blank character in the text portion of the /SSR command. This exit routine allows external subsystem commands to be entered from IMS terminals and Automated Operator Interface (AOI) applications.

IMS passes the command output destination name (LTERM name) to the exit routine. The external subsystem can send a command response to this destination by using the IMS Message Service.

Related Reading: For more information about using the IMS Message Service, refer to “Message Service Exit Routine” on page 543.

For commands from an AOI program or from an input-only device not associated with an output device, the output destination is the IMS MTO; otherwise it is the inputting terminal.

IMS also provides the user ID associated with the command, if any, that the external subsystem might use for security authorization checking.

IMS sends message DFS3612I to the inputting terminal if an /SSR command is entered and a Command exit routine was not provided by the external subsystem.

Activating the Routine

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EEVTP) indicates a control region environment (control region TCB).

The EPL contains:

Offset		
Hex	Decimal	Content
0	0	Address of the parameter count field. The count field contains the value F'5'.
4	4	Address of the EEVT prefix.

Offset Hex	Decimal	Content
8	8	Address of the variable length external subsystem command input. The command input is in the following format:

```
   OFFSET           LENGTH/
 HEX  DEC  NAME   ALIGNMENT      DESCRIPTION

   0    0  MSGLL      2      record length
   2    2  MSGZZ      2      reserved bytes
   4    4  CRC        1      command recognition character
   5    5  CMDDATA    nnn    external subsystem command
```

Offset Hex	Decimal	Content
C	12	Address of the 8-byte alphanumeric destination name (that is, LTERM name) where the command response message, if any, is to be sent. The name is left justified and padded with blanks on the right.
10	16	Address of the 8-character user ID associated with the command input message. The user ID is left justified and padded with blanks on the right. If IMS extended security (SIGNON|SIGNOFF) is not active, or the inputting terminal did not sign on, the user ID field contains the output destination LTERM name.
14	20	Address of the 8-byte RACF group name for the user ID that entered the command. The name is left justified and padded with blanks on the right. The area contains blanks if RACF checking is not in effect.

Contents of Register 15 on Return

Processing of return codes received from the exit routine:

Return Code	Meaning
0	Command exit routine successful. The Command exit routine accepted the command input message.
4	Command exit routine unsuccessful. The Command exit routine rejected the command input message.
20	Should not occur. See “Return Codes” on page 502 for more information.

Commit Continue Exit Routine

The Commit Continue exit routine provides the second phase of the two-phase commit process. In other words, the data associated with the current PSB is committed to the database, locks are released, and cleanup is preformed. This exit routine is activated after all participating subsystems have voted 'yes' (return code 0 from Commit Prepare exit routine) to the commit prepare request. (See “Commit Prepare Exit Routine” on page 508 for more information.)

Activating the Routine

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EEVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Offset Hex	Decimal	Content
0	0	Address of the parameter count field. The count field contains the value F'2'.
4	4	Address of the EEVT prefix.

Offset Hex	Decimal	Content
8	8	Address of the 16-byte recovery token associated with this instance of the transaction. The recovery token identifies the unit of work across one or more subsystems.

Contents of Register 15 on Return

Processing of return codes received from the exit routine:

Return Code	Meaning
0	Commit Continue successful. **Action:** IMS continues normal processing.
4	Commit Continue unsuccessful. ESAP or external subsystem processing of the request failed. **Action:** IMS terminates the application with abend U3046 (the input message is processed; DL/I resources are committed). The control region performs resolve in-doubt processing for the recovery token. (See “Resolve In-Doubt Exit Routine” on page 519 for more information about this exit routine.) The dependent region is terminated, which implicitly terminates the dependent region connection to the external subsystem (Signoff and Terminate Identify exit routines are not called). BMP jobs that must be resubmitted resume processing after the commit point.
20	Should not occur. See “Return Codes” on page 502 for more information.

Commit Prepare Exit Routine

The Commit Prepare exit routine is activated by IMS when an update or non-update transaction reaches a sync point. Sync points include:

- Get unique (GU) call to the message queue
- Application-initiated checkpoint
- Application program termination

Upon return, the exit routine must indicate whether it is prepared to commit all uncommitted changes initiated by the currently scheduled application. The exit routine can indicate whether or not the second phase of the commit process (commit continue) is required. If the transactions associated with the sync point processing are non-update transactions, they do not need to be committed, in which case the exit routine returns with a return code of X'C', requesting that IMS **not** call the Commit Continue exit routine.

Activating the Routine

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EEVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Offset Hex	Decimal	Content
0	0	Address of the parameter count field. The count field contains the value F'2'.

Offset Hex	Decimal	Content
4	4	Address of the EEVT prefix.
8	8	Address of the 16-byte recovery token associated with this instance of the transaction. The recovery token identifies the unit of work across one or more subsystems.

Contents of Register 15 on Return

Processing of return codes received from the exit routine:

Return Code	Meaning
0	Commit Prepare successful. **Action:**IMS continues normal processing.
4	Commit Prepare unsuccessful. The external subsystem is not prepared to perform commit processing at this time. **Action:** • If the application is not terminating, IMS drives the Abort Continue exit routine. An internal ROLB is performed, which returns the input message to the application. • If the sync point was the result of the application terminating, IMS activates the Terminate Thread exit routine with the abort option. (See “Terminate Thread Exit Routine” on page 529 for more information about this exit routine.) The application is terminated with abend U3055, updates are discarded, and the input message is re-enqueued.
8	Commit Prepare unsuccessful. Prepare processing failed in the external subsystem. **Action:** IMS activates the Abort Continue exit routine for all participating subsystems (if the application is not terminating) or the Terminate Thread exit routine with the abort option. The application terminates with abend U3044 and updates are discarded.
C	Commit Prepare successful for nonupdate transactions. **Action:**IMS continues normal processing but does not call the Commit Continue exit routine. The external subsystem indicated that it is processing non-update transactions that do not need to be called for the second phase of commit processing. If the application program is terminating, IMS calls the Terminate Thread exit routine.
18	Commit Prepare unsuccessful. The request was rejected because the recovery token presented by IMS at commit prepare already existed in the external subsystem. One of the following conditions occurred: • Outstanding recovery was not resolved by the Resolve In-doubt exit routine, probably due to errors in the external subsystem. (See “Resolve In-Doubt Exit Routine” on page 519 for more information about this exit routine.) • IMS was cold started and the contents of the recovery token occurred once again. **Action:**IMS pseudo abends the application program with abend U3053 and backs out the previous updates. The application is immediately rescheduled. The dependent region connection is reestablished whereupon a new recovery token is presented to the Signon exit routine.
20	Should not occur. See “Return Codes” on page 502 for more information.

Commit Verify Exit Routine

IMS calls the Commit Verify exit routine during Get Unique (GU) message processing when the transaction is defined as MODE=MULTI. This kind of transaction allows multiple messages to be processed without an intervening commit action.

IMS calls the exit routine before the next message is dequeued and presented to the application program. The exit routine allows the external subsystem to decide if it can properly process a new message without initiating a commit for the previous message. The external subsystem returns to IMS with a return code that requests that IMS continue with normal MODE=MULTI processing or initiate a commit action. If a commit action is requested, IMS will initiate the commit action before dequeuing the next message and will terminate the application program with a “QC” status code.

Activating the Routine

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EEVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Offset		
Hex	**Decimal**	**Content**
0	0	Address of the parameter count field. The count field contains the value F'3'.
4	4	Address of the EEVT prefix.
8	8	Address of the 8-character user ID, left justified and padded with blanks. The user ID is associated with the message that is currently being processed (the next message has not yet been dequeued) and is identical to the one that was presented to the external subsystem at the time IMS last called the Signon exit routine.
C	12	Address of the 16-byte recovery token associated with this instance of the transaction. The recovery token identifies the unit of work across one or more subsystems. This recovery token is identical to the one that was presented to the external subsystem when IMS last called the Signon exit routine.

Contents of Register 15 on Return

Processing of return codes received from the exit routine:

Return Code	**Meaning**
0	Commit Verify processing successful. The external subsystem indicates that it can support MODE=MULTI processing without initiating a commit action. **Action:**IMS continues normal MODE=MULTI processing. The next message will be dequeued and presented to the application program without initiating a commit action.
4	Commit Verify processing successful. The external subsystem indicates that it cannot support MODE=MULTI processing at this time. IMS needs to initiate a commit action. **Action:**IMS terminates the application with a “QC” status and initiates commit processing. Following the commit action, IMS reschedules the application program and the next message is presented for processing.

Return Code	Meaning
8	Commit Verify unsuccessful. Commit Verify processing failed in the external subsystem. **Action:** IMS terminates the application program with abend U3044 and discards all updates.
20	Should not occur. See “Return Codes” on page 502 for more information.

Create Thread Exit Routine

The Create Thread exit routine is activated by IMS to create a thread to the external subsystem. Threads can be created only after the TCB that the application runs under has been identified to the external subsystem. A thread is created for each application that makes a request to the external subsystem. The first request by the application program directed at the selected subsystem initiates the activation. Once the thread is created, application requests flow directly to the external subsystem via the Normal Call exit routine. (See “Normal Call Exit Routine” on page 517 for information about the Normal Call exit routine.)

Activating the Routine

The exit routine is activated in key seven, supervisor state. The EEVT prefix (EEVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Offset		
Hex	**Decimal**	**Content**
0	0	Address of the parameter count field. The count field contains the value F'5'.
4	4	Address of the EEVT prefix.
8	8	Address of the eight-character application program name, left justified and padded with blanks to the right.
C	12	Address of the eight-character PSB name, left justified and padded on the right with blanks.
10	16	Address of the contents of register 0 in the application save area. When register 0 was saved, it contained the address of the external subsystem-directed parameter list constructed by the language interface.
14	20	Address of a two-character transaction characteristic field. The fields are described from left to right. Byte one contains one of the following: **U** Indicates the transaction was defined by the installation as capable of update. **N** Indicates the transaction was defined by the installation as non-update. Byte two contains one of the following: **S** Indicates the transaction was defined by the installation as mode=single. **M** Indicates the transaction was defined by the installation as mode=multiple.

Contents of Register 15 on Return

Processing of return codes received from the exit routine:

Return Code	Meaning
0	Create Thread successful. **Action:** IMS continues normal processing.
4	Create Thread unsuccessful. The external subsystem rejected the specified request. **Action:** IMS activates the Subsystem Not Operational exit routine. The return code from the Subsystem Not Operational exit routine determines further processing. (See “Subsystem Not Operational Exit Routine” on page 524 for more information on this exit routine.) Return code 4, coupled with a return code 4 out of the Subsystem Not Operational exit routine, causes an application loop unless the application checks the return code presented by the API.
8	Create Thread temporarily unsuccessful. The external subsystem was unable to complete the request due to the unavailability of a required resource (resource allocation failure). **Action:**IMS terminates the application program with abend U3048.
C	Create Thread permanently unsuccessful. The request failed in the external subsystem. **Action:** IMS terminates the application program with abend U3045.
10	Create Thread unsuccessful. The request failed in the external subsystem because communications with it are broken. **Action:**IMS terminates the application program with abend U3044.
14	Create Thread unsuccessful. The external subsystem was unable to complete the request due to a definitional conflict. **Action:**IMS terminates the application program with abend U3047.
20	Should not occur. See “Return Codes” on page 502 for more information.
24	Create Thread unsuccessful. A resource deadlock was detected by the external subsystem. **Action:** IMS terminates the application program with abend U777. All changes are backed out and the application is rescheduled.

Echo Exit Routine

The Echo exit routine is activated to determine whether IMS can communicate with the external subsystem. Activation normally occurs after IMS terminates an application program due to an error processing an external subsystem request. The Echo exit routine is expected to send a 'are you there' message or signal to the external subsystem, soliciting a response. This exit routine is called before the Resolve In-Doubt exit routine, described in 519.

Activating the Routine

The exit routine is activated in key seven, supervisor state. The EEVT prefix (EEVTP) indicates a control region environment (control region TCB).

The EPL contains:

Offset Hex	Decimal	Content
0	0	Address of the parameter count field. The count field contains the value F'1'.
4	4	Address of the EEVT prefix.

Contents of Register 15 on Return

Processing of return codes received from the exit routine:

Return Code	Meaning
0	Echo successful. The external subsystem responded to the Echo exit routine indicating it is able to continue communication. **Action:**IMS continues normal application scheduling and processing.
4	Echo unsuccessful. The external subsystem has not responded to the Echo exit routine or responded in error. **Action:**IMS PSTOPs the transaction in question.
20	Should not occur. See “Return Codes” on page 502 for more information.

Identify Exit Routine

IMS activates the Identify exit routine to establish a connection from the control region or a dependent region to the external subsystem. Initial contact from the region to the external subsystem is through this exit routine. (The Identify exit routine is expected to communicate with the external subsystem whereas the Initialization exit routine, if provided, might only perform ESAP initialization and not actually communicate with the external subsystem.) Successful activation of the exit routine (for example, return code 0) is necessary in order for the region to be able to communicate with the external subsystem.

An aspect of the identify concept is the identification of IMS TCBs to the external subsystem. When an IMS TCB terminates abnormally and in some cases when a dependent region terminates normally, IMS does not inform the external subsystem of the termination. The external subsystem should monitor, via MVS end-of-task (EOT) exit routines, the TCBs for the regions that identify so that it can be notified by MVS of a termination that was not communicated by IMS.

In the control region and in an MPP- or IFP-dependent region, the Identify exit routine is activated during region initialization processing unless the Initialization exit routine returned with return code 4 (do not identify). The Identify exit routine (control or dependent region) is also activated when the external subsystem activates (via an exit routine) the Subsystem Startup Service supplied by IMS.

IMS passes a notify message on the control region identify request. If the exit routine returns with return code 4 (notify message accepted), IMS waits for the external subsystem to send the notify message before reactivating the exit routine to establish the connection. This return code is intended to be used (optionally) in the case where the external subsystem is not active when IMS attempts to identify.

Related Reading: See “Notify Message” on page 535 for more information.

Attach Facility

IMS also passes the address of a termination ECB to the control region Identify exit routine. The external subsystem can post this ECB to cause IMS to terminate the connection; for example, when the external subsystem is shutting down.

Related Reading: Refer to “Terminating the External Subsystem Connection” on page 539 for more information.

Activating the Routine from the Control Region

The exit routine is activated in key seven, supervisor state. The EEVT prefix (EEVTP) indicates control region environment (control region TCB).

The exit parameter list (EPL) contains:

Offset		
Hex	**Decimal**	**Content**
0	0	Address of the parameter count field. The count field contains the value F'5'.
4	4	Address of the EEVT prefix.
8	8	Address of the 4-character external subsystem name.
C	12	Address of the 8-character field containing the IMS system ID (4 characters blank filled to 8 bytes). In an XRF complex, this field contains the RSENAME.
10	16	Address of the notify message area. See “Notify Message” on page 535.
14	20	Address of the subsystem termination ECB.

Contents of Register 15 on Return

Processing of return codes received from the exit routine:

Return Code	**Meaning**
0	Identify successful. **Action:**IMS performs resolve in-doubt processing in the control region. (See “Resolve In-Doubt Exit Routine” on page 519).
4	Identify unsuccessful. The external subsystem will send the notify message when it is ready to connect. **Action:**The external subsystem connection is not established. IMS waits for receipt of the notify message before activating the Identify exit routine again. Calling the IMS Subsystem Startup Service after Identify return code 4 does not cause the Identify exit routine to be reactivated.
8	Identify unsuccessful. The notify message was accepted on a previous identify request. **Action:** IMS waits for receipt of the notify message before activating the exit routine again.
C	Identify unsuccessful. The identify process failed, either in the ESAP or the external subsystem. **Action:**If an application is involved, it terminates with abend U3044.
20	Should not occur. See “Return Codes” on page 502 for more information.

Activating the Routine from the Dependent Region

The exit routine is activated in key seven, supervisor state. The EEVT prefix (EEVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Offset Hex	Decimal	Content
0	0	Address of the parameter count field. The count field contains the value F'3'.
4	4	Address of the EEVT prefix.
8	8	Address of the four-character external subsystem name.
C	12	Address of the IMS system ID.

Contents of Register 15 on Return

Processing of return codes received from the exit routine:

Return Code	Meaning
0	Identify successful. The dependent region TCB has successfully identified to the external subsystem. **Action:**Signon processing follows.
C	Identify unsuccessful. The identify process failed, either in the ESAP or the external subsystem. **Action:**If an application is involved, it terminates with abend U3044.
20	Should not occur. See “Return Codes” on page 502 for more information.

Initialization Exit Routine

IMS activates the optional Initialization exit routine to allow the ESAP to initialize work areas or control blocks in the following instances:

- During the initial stages of establishing a connection from the control or dependent regions. Activation occurs after IMS has constructed its required control blocks as well as the control blocks for the external subsystem. This action takes place before the control or dependent regions have their respective Identify exit routine activated.
- In a dependent region after an application program abend.

If the Initialization exit routine sets the 'do not identify' return code (return code 4), or if an Initialization exit routine is not supplied, IMS does not automatically perform identify processing for the region. “Initiating the External Subsystem Connection” on page 533 describes how the identify is eventually performed.

Activating the Routine from the Control Region

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EEVTP) indicates a control region environment (control region TCB).

The EPL contains:

Offset Hex	Decimal	Content
0	0	Address of the parameter count field. The count field contains the value F'2'.
4	4	Address of the EEVT prefix.

Offset Hex	Decimal	Content
8	8	Address of the 1-byte alphabetic region error option (REO) character defined by the installation. The exit routine should save the error option for future reference when a decision concerning the application is required.

Contents of Register 15 on Return

Processing of return codes received from the exit routine:

Return Code	Meaning
0	Initialization successful. **Action:**IMS attempts to initiate a connection with the external subsystem by activating the Identify exit routine.
4	Initialization successful. Do not identify to the external subsystem. **Action:**IMS does not perform identify processing during control region initialization. It is now the responsibility of the external subsystem to initiate the connection using the IMS Subsystem Startup Service (see “Subsystem Startup Service Exit Routine” on page 545).
8	Initialization unsuccessful. **Action:**IMS does not initiate a connection to the subsystem. The external subsystem is marked as unstartable. The /START SUBSYS command resets the condition.
20	Should not occur. See “Return Codes” on page 502 for more information.

Activating the Routine from the Dependent Region

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EEVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Offset Hex	Decimal	Content
0	0	Address of the parameter count field. The count field contains the value F'2'.
4	4	Address of the EEVT prefix.
8	8	Address of the 1-byte alphabetic region error option character. The region error option is user-defined as part of the SSM.PROCLIB member. The exit routine code should save the error option for future reference when a decision concerning the application is required.

Contents of Register 15 on Return

Processing of return codes received from the exit routine:

Return Code	Meaning
0	Initialization successful. **Action:**For an MPP or IFP region, IMS initiates a connection to the external subsystem during region initialization. (IMS does not automatically initiate a connection for a BMP region. See the description for return code 4.)

Return Code	Meaning
4	Initialization successful. Do not identify to the external subsystem. **Action:**IMS does not automatically initiate a connection for the dependent region. When the region processes the first application call to the external subsystem, the ESAP is expected to activate the IMS Subsystem Startup Service (from the Subsystem Not Operational exit routine, see "Subsystem Not Operational Exit Routine" on page 524) to initiate the connection. This is always the case for BMP regions (that is, when return code 0 is set). Return code 4 has significance only for MPP and IFP regions.
8	Initialization unsuccessful. **Action:**IMS does not initiate a connection to the subsystem for the life of the execution of this dependent region.
20	Should not occur. See "Return Codes" on page 502 for more information.

Normal Call Exit Routine

The subsystem-supplied Normal Call exit routine is activated by IMS when a subsystem-directed request is made by an application program and a thread to the subsystem is present. It is the responsibility of the Normal Call exit routine to:

- Communicate normal call and data to the external subsystem.
- Handle responses.
- Supply status codes to the application program.

Activating the Routine

The exit routine is activated in the caller's key. The caller is IMS, an application program, or an external subsystem-supplied exit routine, and is either authorized or unauthorized. If the caller is authorized, the exit routine is activated in key 7, supervisor state. If the caller is unauthorized, the exit routine is activated in key 8, problem program state. The EEVT prefix (EEVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Offset		
Hex	Decimal	Content
0	0	Address of the parameter count field. The count field contains the value F'6'.
4	4	Address of the EEVT prefix.
8	8	Address of the contents of register 0 in the application save area. At this time register 0 contains the address of the external subsystem-directed parameter list as constructed by the language interface.
C	12	Address of the contents of register 1 in the application save area. Register 1 contains the address of the application parameter list.
10	16	Address of the 16-byte recovery token associated with this instance of the transaction. The recovery token identifies the unit of work across one or more subsystems.

Offset Hex	Decimal	Content
14	20	Address of a one-character field which identifies the authorization state: **A** The caller is authorized and the exit routine is activated in key seven, supervisor state. **U** The caller is unauthorized and the exit routine is activated in key eight, problem program state.
18	24	Address of a 12-word buffer area provided for specific language function calls. External subsystems that require IMS to call internal exit routines for post-normal call processing can use this buffer to pass data to the internal exit routine. If post-normal call processing is required, IMS passes the address of the buffer to the associated internal exit routine. If post-normal call processing is **not** required, the external subsystem should not use this parameter. For more information, see return code 12. **Restriction:** The use of this address field is restricted to those external subsystems that have negotiated the definition and use of an internal exit routine for post-normal call processing. Requests for this support should be made through the IBM User Requirements Process, which includes GUIDE, SHARE, and vendor requirements processing.

Contents of Register 15 on Return

Processing of return codes received from the exit routine:

Return Code	Meaning
0	Normal Call successful. **Action:**IMS continues normal processing.
4	Normal Call unsuccessful. A resource deadlock is detected by the external subsystem. **Action:**IMS terminates the application program with abend U777. All changes are backed out and the application is rescheduled.
8	Normal Call unsuccessful. A failure in the external subsystem occurred while processing the request. **Action:**IMS terminates the application with abend U3044.
C	Normal Call successful. The external subsystem requested the scheduling of an associated internal exit routine for post-normal call processing. IMS calls the internal exit routine before returning control to the application program. **Action:**If there is an internal exit routine associated with the language function call, IMS calls the associated internal exit routine and passes the address of the buffer. If there are no internal exit routines associated with it, IMS ignores the request and performs the processing associated with return code 0. The actual interface to an internal exit routine is unique to that routine and depends on the type of external subsystem. The external subsystem-specific interfaces are not documented.
20	Should not occur. See “Return Codes” on page 502 for more information.

Resolve In-Doubt Exit Routine

The Resolve In-Doubt exit routine provided by the external subsystem aids in the coordination of recovery between the two subsystems. IMS, as the recovery coordinator, always calls this exit routine after successful completion of the identify process. IMS indicates in the EPL whether or not recovery must take place for the units of work in question.

The Resolve In-Doubt exit routine is activated once for each outstanding IMS recovery token. It is called after the Echo exit routine, discussed on page512, is called. The external subsystem directs IMS to save or destroy the recovery token. More information on exit routine responses is in “Contents of Register 15 on Return” on page 520.

The Resolve In-Doubt exit routine has the option of allowing the two subsystems to continue communication with or without all recovery tokens resolved. If communication continues and outstanding recovery tokens exist, an authorized operator may direct IMS to delete its recovery tokens using the /CHANGE command.

Related Reading: Refer to *IMS/ESA Operator’s Reference*for information on the /CHANGE command.

If the Resolve In-Doubt exit routine address is not present in the EEVT and outstanding recovery tokens do not exist, IMS allows the connection process to continue. However, if a recovery token does exist, IMS terminates the connection and informs the MTO with message DFS3602I.

The Resolve In-Doubt exit routine is also activated after the abend of an application program that had a connection (thread) to the external subsystem.

Activating the Routine

The exit routine is activated in key seven, supervisor state. The EEVT prefix (EEVTP) indicates a control region environment (control region TCB).

The EPL contains:

Offset		
Hex	**Decimal**	**Content**
0	0	Address of the parameter count field. The count field contains the value F'4'.
4	4	Address of the EEVT prefix.

Offset Hex	Decimal	Content
8	8	Address of a two-character field: • Byte 1 contains an indicator, either 'C' or 'W', on the first activation of the Resolve In-Doubt exit routine during the current IMS execution. On subsequent activations, the byte contains binary zeroes. (For example, if the external subsystem terminates and restarts while IMS remains active, when the connection is reestablished, the byte will contain binary zeroes.) **C** Indicates IMS was cold started. All subsequent fields in the parameter list contain binary zeroes. **W** Indicates IMS was warm started. • Byte 2 is set to 'L' after the last recovery token of the current sequence is processed. For all other activations, the byte is set to binary zeroes. **L** Indicates either that there are no recovery tokens to be resolved, or that all recovery tokens that were to be resolved at this time have been processed. If 'L' is not set on an activation of the exit routine, the exit routine should expect to be activated one or more times (once per recovery token) until 'L' is set. A recovery token is not passed on the last ('L') activation. When 'C' is set in byte one, 'L' is always set because IMS does not save recovery information across a cold start.
C	12	Address of a two-character field indicating whether the recovery unit should be aborted or committed. **CO** Commit **AB** Abort
10	16	Address of the 16-byte recovery token associated with this instance of the transaction. The recovery token identifies the unit of work across one or more subsystems.

Contents of Register 15 on Return

Processing of return codes received from the exit routine:

Return Code	Meaning
0	Resolve In-Doubt successful. **Action:**IMS continues normal processing. The recovery token is destroyed.
4	Resolve In-Doubt unsuccessful. Return code 4 should be set when the external subsystem chooses not to resolve (commit or abort as directed) the unit of work at this time and expects IMS to pass the recovery token for resolve in-doubt processing at a later time. This return code is not intended for the case where resources have become inconsistent or the possibility exists (see return code C). **Action:**IMS saves the recovery token. The connection status remains unchanged. IMS assumes that the indicated unit of work is in in-doubt status in the external subsystem (for example, resources have not become inconsistent). The saved recovery token will be included in the resolve in-doubt processing for the next establishment of the connection. IMS does not inform the installation that the unit of work was not resolved.

Return Code	Meaning
8	Resolve In-doubt unsuccessful. Return code 8 can be used when the external subsystem chooses not to resolve the unit of work during exit processing but saves the commit direction so that IMS does not need to save the recovery token. This return code is not intended for the case where resources have become inconsistent (see return code C). Return code 8 might be used when the indicated unit of work is not in in-doubt status in the external subsystem but resource consistency is not jeopardized, however, **caution is advised**. External subsystem-specific processing that is not coordinated with IMS can result in IMS holding a recovery token in in-doubt status when the unit of work is not in-doubt in the external subsystem (for example, external subsystem “cold start”, or manual recovery of a unit of work by the installation if allowed by the external subsystem). If the external subsystem can determine that its prior resolution of a unit of work (explicit or implicit) agrees with the commit direction passed to the exit routine, return code 8 can be set; otherwise, return code C should be set. **Action:**IMS destroys the recovery token. The connection status remains unchanged (IMS assumes that resource consistency is maintained). The IMS action is the same as for return code 0. Setting return code 8 allows for an audit trail of the “not-quite-normal” cases.
C	Resolve In-doubt unsuccessful. Return code C should be used when resources in IMS and the external subsystem have become inconsistent or when the possibility exists. It is intended to be used, for example, when the recovery token passed to the exit does not exist (is unknown) in the external subsystem, or when the external subsystem has in-doubt recovery tokens remaining when IMS has finished processing its in-doubt list. The external subsystem should take appropriate additional action to maintain integrity and assist the installation in resolving resource inconsistencies. **Action:**IMS terminates the connection and saves all remaining recovery tokens. IMS also issues message DFS3602I to notify the installation of a resource problem. • If a recovery token was passed on the exit routine activation (for example, **L** was not set), IMS terminates the connection to the external subsystem. The recovery token passed and all remaining recovery tokens are saved. • If this is the last activation (**L** was set), the connection status is unchanged. Dependent regions are allowed to connect to the external subsystem.
20	Should not occur. See “Return Codes” on page 502 for more information.

Signoff Exit Routine

The Signoff exit routine is activated by IMS when:

- IMS is shutting down.
- The external subsystem activates the IMS Subsystem Termination Service.
- The subsystem termination notification ECB is posted.
- The `/STOP SUBSYS` command is entered and IMS is terminating all the subsystem connections.

IMS attempts to activate the exit routine as part of an orderly/catastrophic shutdown process of the two subsystems.

Activating the Routine

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EEVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Offset		
Hex	**Decimal**	**Content**
0	0	Address of the parameter count field. The count field contains the value F'1'.
4	4	Address of the EEVT prefix.

Contents of Register 15 on Return

Processing of return codes received from the exit routine:

Return Code	**Meaning**
0	Signoff successful.
8	Signoff unsuccessful. The ESAP or external subsystem processing of the request failed. Action: IMS terminates the dependent region connection with the external subsystem, allowing other dependents to continue processing.
20	Should not occur. See “Return Codes” on page 502 for more information.

Signon Exit Routine

The Signon exit routine informs the external subsystem of the user ID associated with the transaction input message. The user ID can be:

- The inputting LTERM name (if the terminal user is not signed on)
- The RACF/user-authorized user ID associated with a non-message driven BMP
- The PSB name specified on the job card of a non-message driven BMP

For a non-message driven BMP, IMS examines the ASXBUSER field in the ASCB extension. If the field does not contain binary zeroes or blanks, IMS assumes an authorized user ID is present and passes it to the Signon exit routine. The ID is extracted from the 7-byte ASXBUSER field and inserted into the parameter list, left justified and padded with blanks to the right. If the field does contain binary zeroes or blanks, IMS passes the PSB name as indicated above.

When a connection is initially established, the Signon exit routine is activated before a thread is created by the Create Thread exit routine. All subsequent requests result in the exit routine being activated after a thread is created. For example, Signon is activated for each message processed during a single scheduling, whether or not the messages are separated by commit processing.

Activating the Routine

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EEVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Offset Hex	Decimal	Content
0	0	Address of the parameter count field. The count field contains the value F'5'.
4	4	Address of the EEVT prefix.
8	8	Address of the eight-character user ID, left justified and padded on the right with blanks.
C	12	Address of the 16-byte recovery token associated with this instance of the transaction. The recovery token identifies the unit of work across one or more subsystems.
10	16	Address of the 8-byte RACF group name for the user ID that entered the transaction. The name is left justified and padded with blanks on the right. The area contains blanks if RACF checking is not in effect.
14	20	Address of the field containing the performance block token for MVS workload management support.
18	24	Address of the XID token associated with this transaction. The XID token will identify participants in a Distributed Syncpoint environment.

Contents of Register 15 on Return

Processing of return codes received from the exit routine:

Return Code	Meaning
0	Signon successful. **Action:**IMS continues normal processing.
4	Signon unsuccessful. The external subsystem rejected the specified request. **Action:**IMS activates the Subsystem Not Operational exit routine. (See “Subsystem Not Operational Exit Routine” on page 524 for more information on this exit routine.) The return code from the Subsystem Not Operational exit routine determines further processing.
8	Signon temporarily unsuccessful. The external subsystem was unable to complete the request due to the unavailability of a required resource (resource allocation failure). **Action:**IMS terminates the application program with abend U3048.
C	Signon permanently unsuccessful. The request failed in the external subsystem. **Action:** IMS terminates the application program with abend U3045.
10	Signon unsuccessful. The request failed in the external subsystem because communications with it are broken. **Action:**IMS terminates the application program with abend U3044.
14	Signon unsuccessful. The external subsystem was unable to complete the request due to a resource definitional conflict between the two subsystems. **Action:**IMS terminates the application program with abend U3047.

Return Code	Meaning
18,1C	Signon unsuccessful. The request was rejected because the recovery token presented by IMS at signon already exists in the external subsystem. One of save following conditions occurred: • Outstanding recovery was not resolved by the Resolve In-doubt exit routine, probably due to errors in the external subsystem. (See “Resolve In-Doubt Exit Routine” on page 519 for more information on this exit routine.) • IMS was cold started and the contents of the recovery token occurred once again. **Action:**IMS pseudo abends the application program with abend U3053 and backs out the previous updates. The application is immediately rescheduled. The dependent region connection is re-established whereupon a new recovery token is presented to the Signon exit routine.
20	Should not occur. See “Return Codes” on page 502 for more information.

Subsystem Not Operational Exit Routine

The Subsystem Not Operational exit routine is viewed as a utility type of exit routine. It is typically activated when IMS encounters an unusual situation. IMS activates this exit routine when:

- An application program directs a request to the external subsystem and a connection does not exist. The Subsystem Not Operational exit routine can activate the IMS Subsystem Startup Service exit routine to initiate a connection.
- Return code 4 is returned from the Signon or Create Thread exit routines. (See “Signon Exit Routine” on page 522 and “Create Thread Exit Routine” on page 511 for more information about these exit routines.)
- The external subsystem tells IMS it is quiescing before creation of a thread.

Activating the Routine

The exit routine is activated in the caller’s key. The caller is IMS, an application program, or an external subsystem-supplied exit routine, and is either authorized or unauthorized. If the caller is authorized, the exit routine is activated in key seven, supervisor state. If the caller is unauthorized, the exit routine is activated in key eight, problem program state. The EEVT prefix (EEVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Offset		
Hex	**Decimal**	**Content**
0	0	Address of the parameter count field. The count field contains the value F'10'.
4	4	Address of the EEVT prefix.
8	8	Address of the contents of register 0 in the application program save area. At this time register 0 contains the address of the external subsystem-directed parameter list as constructed by the language interface.
C	12	Address of the contents of register 1 in the application program save area. Register 1 contains the address of the application parameter list.

Offset Hex	Decimal	Content
10	16	Address of a one-character information field. The contents indicate why the Subsystem Not Operational exit routine is being activated. The field contains: **A** Return code 4 was returned by the Signon or Create Thread exit routines. (See “Signon Exit Routine” on page 522 and “Create Thread Exit Routine” on page 511 for more information about these exit routines.) **C** The IMS control region has not identified to the external subsystem. This condition was discovered when an application directed a request to the subsystem. The Subsystem Not Available exit routine can activate the IMS Subsystem Startup Service to initiate a connection. **D** An application issued a call to the external subsystem but the dependent region has not identified. The Subsystem Not Operational exit routine can activate the IMS Subsystem Startup Service to initiate a connection. **Q** The external subsystem notified IMS that it is terminating in a quiesce fashion. Prior to the creation of a thread is the only interval where 'Q' is passed to the external subsystem (applicable mainly when the region error option (REO) is an 'R'). **T** The external subsystem notified IMS that it is either abnormally terminating or terminating in a quick fashion. It is highly likely that a subsystem-directed request will fail. IMS notifies the external subsystem when servicing a subsystem request (such as Create Thread).
14	20	Address of a one-character default application option field. This field contains the region error option (REO) defined by the installation in the external subsystem PROCLIB member to take effect in the event an application issues a subsystem-directed request when a complete authorized connection does not exist. This field is always valid. If an option is not specified, IMS uses its elected default ('R').
18	24	Address of a four-byte character format field containing the name of the subsystem associated with the request. The name is left justified and padded with blanks.
1C	28	Address of a fullword where the Subsystem Not Operational exit routine optionally returns an MVS format abend code.
20	32	Address of the application program name scheduled at this time. The name is assumed to be left justified and padded with blanks to the left. The maximum length is eight bytes.
24	36	Address of the 16-byte recovery token associated with this instance of the transaction. The recovery token identifies the unit of work across one or more subsystems.
28	40	Address of a one-character field which identifies the authorization state: **A** The caller is authorized and the exit routine is activated in key seven, supervisor state. **U** The caller is unauthorized and the exit routine is activated in key eight, problem program state.

Contents of Register 15 on Return

Processing of return codes received from the exit routine:

Return Code	Meaning
0	Subsystem Not Operational call successful. **Action:**IMS continues normal processing. Normal processing in this case means activating the Signon and Create Thread exit routines to complete the external subsystem connection. (See "Signon Exit Routine" on page 522 and "Create Thread Exit Routine" on page 511 for more information about these exit routines.)
4	Subsystem Not Operational call successful/unsuccessful. This exit routine elected to pass status feedback to the application. **Action:**IMS returns control to the application program. The application is permitted to continue running. A loop between the Create Thread exit routine and the Subsystem Not Operational exit routine might result if the application program does not check for nonzero return codes from the API.
8	Subsystem Not Operational call unsuccessful. **Action:**IMS terminates the application program with abend U3044. The transaction input is saved and all uncommitted changes are backed out. The dependent region remains available for application processing.
C	Subsystem Not Operational call unsuccessful. **Action:**IMS terminates the application program with abend U3047 and discards the input. The dependent region remains available for application processing.
10	Subsystem Not Operational call unsuccessful. **Action:**IMS uses the MVS format abend code returned by this exit routine to abend the application.
20	Should not occur. See "Return Codes" on page 502 for more information.

Subsystem Termination Exit Routine

The Subsystem Termination exit routine is activated in the control region when IMS or the external subsystem terminate; activation follows the Terminate Identify exit routine. It is used for cleaning up work areas or freeing memory.

The Subsystem Termination exit routine should execute in parallel with normal and abnormal IMS or external subsystem termination processing. External subsystem termination is recognized when the subsystem posts the termination ECB.

Activating the Routine

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EEVTP) indicates a control region environment (control region TCB).

The EPL contains:

Offset		
Hex	**Decimal**	**Content**
0	0	Address of the parameter count field. The count field contains the value F'2'.
4	4	Address of the EEVT prefix.

Offset Hex	Decimal	Content
8	8	Address of a 1-byte character format field indicating the reason for subsystem termination. The field contains one of the following:

A	IMS is shutting down in a normal fashion (`/CHECKPOINT FREEZE`). IMS makes sure new connections are not established and permits existing ones to terminate normally.
B	IMS is shutting down abnormally (abend). Some abend conditions might prohibit the activation of this exit routine.
C	The external subsystem notified IMS that it is terminating in a quiesce fashion. IMS makes sure new connections are not established and permits existing ones to terminate normally.
D	The external subsystem notified IMS that it is terminating abnormally (catastrophic). IMS makes sure new connections are not attempted and terminates existing ones.
E	The connection between the subsystems is being quiesced by IMS. IMS is not shutting down but remains available. The termination of the connection is the result of the `/STOP` command, a bad return code from an exit routine, or a required exit routine missing.
F	The connection between the subsystems is being terminated because the IMS Subsystem Termination Service exit routine was activated by an external subsystem exit routine.

Contents of Register 15 on Return

Processing of return codes received from the exit routine:

Return Code	Meaning
0	Subsystem Termination successful. **Action:**IMS continues subsystem termination.
8	Subsystem Termination unsuccessful. The ESAP or the external subsystem encountered a failure in processing the termination notification. **Action:**IMS continues termination processing. Future connection requests are honored.
20	Should not occur. See “Return Codes” on page 502 for more information.

Terminate Identify Exit Routine

Each IMS region that has an external subsystem connection must first identify to the subsystem. Identify must first be complete for the control region before any dependent regions identify. This hierarchical structure allows the control region to act as recovery coordinator for the dependent regions. If a dependent region were to fail, the control region intervenes and instructs the external subsystem to commit or abort the dependent region units of work.

The Terminate Identify exit routine is activated by IMS to terminate this hierarchical structure when:

- IMS is shutting down.
- The subsystem activates the IMS Subsystem Termination Service.

- The subsystem termination notification ECB is posted.
- The /STOP SUBSYS command is entered.

Activation is part of an orderly/catastrophic shutdown or disconnecting process of the two subsystems.

Activating the Routine from the Control Region

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EEVTP) indicates a control region environment (control region TCB).

The EPL contains:

Offset Hex	Decimal	Content
0	0	Address of the parameter count field. The count field contains the value F'1'.
4	4	Address of the EEVT prefix.

Contents of Register 15 on Return

Processing of return codes received from the exit routine:

Return Code	Meaning
0	Terminate Identify successful. **Action:**IMS continues termination processing.
8	Terminate Identify unsuccessful. The ESAP or external subsystem processing of the request failed. **Action:** IMS continues connection termination processing without affecting other dependent region connections.
20	Should not occur. See “Return Codes” on page 502 for more information.

Activating the Routine from the Dependent Region

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EEVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Offset Hex	Decimal	Content
0	0	Address of the parameter count field. The count field contains the value F'1'.
4	4	Address of the EEVT prefix.

Contents of Register 15 on Return

Processing of return codes received from the exit routine:

Return Code	Meaning
0	Terminate Identify successful. **Action:**IMS continues termination processing.

Return Code	Meaning
8	Terminate Identify unsuccessful. The ESAP or external subsystem processing of the request failed. **Action:**IMS continues connection termination processing without affecting other dependent region connections.
20	Should not occur. See “Return Codes” on page 502 for more information.

Terminate Thread Exit Routine

The Terminate Thread exit routine disconnects the application from the external subsystem. It is activated when the application program terminates normally.

The second phase of the commit processing at application termination is done through the Terminate Thread exit routine. Therefore, a commit option is specified on the activation. The options are described under “Activating the Routine”.

Activating the Routine

The exit routine is activated in key 7, supervisor state. The EEVT prefix (EVVTP) indicates a dependent region environment (dependent region TCB).

The EPL contains:

Hex	Decimal	Content
0	0	Address of the parameter count field. The count field contains the value F'3'.
4	4	Address of the EEVT prefix.
8	8	Address of the four-byte commit option character string: **COMM** Commit and terminate the thread. (Normal application termination is an example of when this option is set). **ABRT** Abort and terminate the thread. **DEAL** Deallocate resources with terminate thread (nothing to commit).
C	12	Address of the 16-byte recovery token associated with this instance of the transaction. The recovery token identifies the unit of work across one or more subsystems.

Contents of Register 15 on Return

Processing of return codes received from the exit routine:

Return Code	Meaning
0	Terminate Thread successful. **Action:**IMS continues normal processing.

Return Code	Meaning
4	Terminate Thread unsuccessful. The ESAP or external subsystem processing of the request failed. **Action:**IMS terminates the application with an abend. The dependent region connection to the external subsystem is also terminated. Resolve in-doubt processing for the recovery token is performed in the control region. (See “Resolve In-Doubt Exit Routine” on page 519 for more information about this exit routine.) • For the COMM (commit) option: The application is terminated with abend U3046 (the input message is processed; DL/I resources are committed). BMP jobs must be resubmitted; they resume processing after the commit point. • For the ABRT (abort) option: The application is terminated with abend U3045 (the input message is deleted; DL/I resources are backed out). BMP jobs must be resubmitted; they resume processing at the prior commit point.
8	Terminate Thread unsuccessful. The Terminate Thread exit routine has either detected an error with the request information or considers the request invalid at this time. This return code should only be used when the commit option character string is 'DE'. **Action:**IMS continues normal processing.
20	Should not occur. See “Return Codes” on page 502 for more information.

Chapter 76. IMS Attach Facility Processing

This chapter describes the processing that the IMS External Subsystem Attach Facility does during control region initialization.

Loading the External Subsystem Attachment Package

During control region initialization, IMS loads external subsystem-supplied tables using the table names specified by the installation (in the external subsystem definition member in IMS.PROCLIB). IMS loads the external subsystem module table (ESMT) and then loads the external subsystem modules defined in the table. The resource translation table (RTT) is also loaded, if provided. If an error occurs during this process, IMS puts the subsystem in 'stopped' status and does not establish a connection. However, IMS will reaccess the definition (PROCLIB) and reattempt this process if a `/START SUBSYS` command is received. Possible errors are:

- Unable to process the PROCLIB member
- Unable to open the external subsystem load library
- Unable to load the ESMT (incorrect name specified, not in library)

IMS stores the addresses of the ESMT and the RTT in the EEVTP control block fields EEVPESMT and EEVPRTTA, respectively.

Creating the EEVT Control Block

IMS creates and initializes the EEVT control block based on information contained in the ESMT. (The IMS DFSEMODL macro is used to generate the ESMT. ESMT generation is described in “Chapter 74. Creating the External Subsystem Module Table” on page 493.) The size and subpool for the EEVT storage request are obtained from the ESMT. The EEVT is created in subpool 230 unless subpool 251 is specified in the ESMT. The external subsystem must ensure that the size specified for the EEVT is at least as large as the size indicated in the IMS EEVT mapping.

IMS stores the EEVT address into the EEVTP control block based on an offset specified in the ESMT. The external subsystem must ensure, therefore, that the offset generated in the ESMT (using the DFSEMODL macro) points to the EEVPEEA field in the IMS EEVTP mapping.

IMS does not check whether the EEVT pointer field, EEVPEEA, in the EEVTP is initialized by this process. In fact, the offset in the ESMT could cause IMS to store the address into some other field in the EEVTP designated for some other use, possibly causing a problem. Thus, **the external subsystem must ensure that the correct offset is generated into the ESMT**.

This process allows the external subsystem to specify another set of modules for IMS to load (IMS would not activate these modules). Both lists of module addresses, one being the EEVT, would be anchored in the EEVTP.

IMS does not use the EEVTLDIR field. Actually more than two sets of modules could be defined in the ESMT (subtables) and loaded by IMS except that there aren’t enough fields in the EEVTP to anchor the address lists.

Loading External Subsystem Modules

As IMS loads external modules defined in the ESMT, the module addresses are stored in the EEVT. The module definitions provide the offsets to the locations in the EEVT for the addresses. IMS does not check whether or not required exit routine addresses have been set by the module loading process. If the external subsystem chooses, the ESAP can set exit routine addresses in the EEVT once IMS has passed control to it. For example, the external subsystem can provide multiple exit routines in one load module and have the ESAP set the individual exit routine module addresses.

Whether through module definitions in the ESMT or through ESAP processing, the external subsystem must ensure that the address of an exit routine is present in the EEVT when IMS needs to activate the exit routine. Exit routine addresses must be placed in the EEVT according to the IMS EEVT mapping.

Some of the exit routines prescribed by IMS are activated only in the control region; some are activated only in the dependent regions. The external subsystem can indicate, in the module definition, if a module is not to be loaded in the control region or in dependent regions. Exit routine module definitions should specify loading according to the following:

- Exit routines activated in the control and dependent regions:
  - Identify
  - Initialization
  - Terminate Identify
- Exit routines activated only in the control region:
  - Command
  - Echo
  - Resolve In-doubt
  - Subsystem Termination
- Exit routines activated only in dependent regions:
  - Abort Continue
  - Commit Continue
  - Commit Prepare
  - Commit Verify
  - Create Thread
  - Normal Call
  - Signoff
  - Signon
  - Subsystem Not Operational
  - Terminate Thread

In the control region, IMS loads external subsystem modules into the subpool(s) specified in the module definitions. If subpool 251 is specified for a module in dependent regions, IMS loads the module in subpool 251; otherwise, it is loaded in subpool 230.

An external subsystem uses only those exit routines that it needs to communicate with IMS, although some exit routines are required and others are optional. When a required exit routine does not exist, IMS generates an error message when it tries to call the exit routine and terminates the connection with the external subsystem.

If your external subsystem does not need the function that an exit routine is designed to perform, you can write the exit routine so that one exists when IMS calls it but that so no operations are performed. (An exit routine can contain common code, such as SR 15,15 and BR 14 logic, which ESS branches to when the exit routine is called and which doesn't perform any specific operation.) During processing of the Initialization exit routine, the external subsystem can update the addresses in the DFSEEVT DSECT (from both the control region and dependent regions, if necessary) and point to these exit routines. This action allows IMS to function normally yet not issue error messages and terminate an external connection if an exit routine does not exist.

Creating Work Areas for the ESAP

After loading the ESAP modules, IMS obtains work area storage for the ESAP if work area definitions are contained in the ESMT.

Related Reading: The IMS DFSEWAL macro is used to generate work area definitions in the ESMT. For more information, refer to “Chapter 74. Creating the External Subsystem Module Table” on page 493.

The process IMS uses to create the work areas is similar to the process used to load ESAP modules except that IMS can either:

- Create the control block for the work area addresses (as it creates the EEVT for the ESAP module addresses), or
- Store the work area addresses into the same control block that has the module addresses.

The intended use of the EEVPEWA field in the EEVTP is to hold the address of a control block referred to as the external subsystem work area list (EWAL) that contains the addresses of the work areas created for the ESAP. However, the external subsystem must have provided the appropriate specifications in the ESMT to cause IMS to store the address of the EWAL in this field.

Related Reading: To see how this can be done, refer to the discussion in “Chapter 74. Creating the External Subsystem Module Table” on page 493.

IMS creates each work area either in the control region or in dependent regions or both, according to the definition. Storage is obtained in subpool 251, if specified; otherwise it is obtained in subpool 230.

Initiating the External Subsystem Connection

IMS automatically connects to the external subsystem during control region initialization processing (for example, when IMS is started) unless the external subsystem chooses to defer the control region identify to a later time.

The external subsystem defers the connection by returning from the control region Initialization exit routine with return code 4 (do not identify), or by not providing an Initialization exit routine for the control region. Use of this option is further discussed under “Deferring the Control Region Identify” on page 534.

If the external subsystem uses the notify message mechanism provided by IMS (and if the external subsystem is not up when IMS activates the Identify exit routine) the connection is automatically established when the external subsystem is started. Return Code 4 from the Identify exit routine causes IMS to wait for the external subsystem to send the notify message passed to the exit routine, and to reactivate the exit routine when the message is received. “Notify Message” on page 535 further describes the use of the notify mechanism.

If the notify mechanism is not used, the Identify exit routine should return with return code 12, in which case the connection is put in stopped status. Stopped status must be removed by a `/START SUBSYS` command before IMS will establish a connection.

Once the connection has been established, IMS performs Resolve In-doubt processing to resolve any outstanding recovery tokens with the external subsystem. If outstanding recovery tokens exist and a Resolve In-doubt exit routine was not supplied, IMS terminates and stops the connection; otherwise dependent regions are allowed to connect to the external subsystem.

Deferring the Control Region Identify

The external subsystem can defer the control region identify if it prefers to have the connection established at some later time. IMS schedules and gives control to application programs whether or not a connection exists to the external subsystem. The external subsystem thus could choose to wait until an application program call has to be serviced (first call for its resources) before connecting to IMS. Of course, to process calls, the control and dependent region connections and the application thread must exist.

When the control region identify is deferred, the identify is done when:

- The external subsystem, via an exit routine, activates the IMS Subsystem Startup Service.
- An MPP or IFP dependent region that can access the external subsystem is started.
- A `/START SUBSYS` command naming the external subsystem is processed.

Using the IMS Subsystem Startup Service

When the control region identify is deferred, the external subsystem can activate the IMS Subsystem Startup Service when it wants the control region Identify exit routine to be driven. To be more specific, if a connection does not exist when the first application call for external subsystem services is processed by a dependent region, IMS does not automatically attempt to identify. The external subsystem must activate the Startup Service to establish the connection (if it **wants** to process application calls).

As discussed under “Establishing Dependent Region Connections” on page 535, the Startup Service is also used to establish dependent region connections.

When an external subsystem call from an application is processed before the control region or dependent region has been identified to the external subsystem, the dependent region activates the Subsystem Not Operational exit routine. The external subsystem is expected to call the Subsystem Startup Service from this exit routine to establish the connection.

When activated, the Startup Service establishes the control and dependent region connections, if the control region identify has not been done. If the control region identify has been done, it establishes only the dependent region connection. If IMS is waiting for the external subsystem to send the notify message, which it accepted on a previous Identify exit routine invocation, the Startup Service returns an error return code and does not establish the connection. For details on activating the Startup Service, see “Subsystem Startup Service Exit Routine” on page 545.

Establishing Dependent Region Connections

MPP and IFP Regions

Identify processing for an MPP or an IFP dependent region is similar to identify processing for the control region in that the dependent region automatically activates its Identify exit routine to establish a connection during dependent region initialization, unless the external subsystem defers the identify.

If the control region connection has not been established when the dependent region would automatically identify, IMS attempts to identify the control region. If successful, the dependent region identify is performed. Thus, if the control region identify is deferred but dependent regions are allowed to connect automatically, the control region Identify exit routine might be activated (automatically) when a dependent region is started.

When the identify for a dependent region is deferred, the connection to the external subsystem is established when the first application program call to the external subsystem is issued in the region. In this case, connection processing is the same as for a BMP dependent region.

BMP Regions

The connection from a BMP dependent region is not established until the first application call to the external subsystem is processed by the region. The connection is automatically established.

Return code 4 from the Initialization exit routine (deferred identify) for a BMP region has no effect.

Notify Message

IMS passes the address of a notify message on the Identify exit routine invocation for the control region. If the external subsystem is not active (has not been started), the Identify exit routine can indicate (return code 4) to IMS that the notify message has been accepted and will be sent to IMS when the external subsystem is active. The external subsystem, once started, sends the message to IMS via an internal MVS `MODIFY` command (SVC 34) to alert IMS that it is ready to connect. On receipt of the notify message, IMS reactivates the Identify exit routine to establish the connection.

External subsystem code that is always present in the MVS system (*early code*), for example, might be used as the means to pass the notify message to the external subsystem. The Identify Exit queues the message to the early code so that it is available to the external subsystem whenever it is started.

IMS passes the notify message to the Identify exit routine in the following format:

LL	ZZ	message_text
2	2	variable

where:

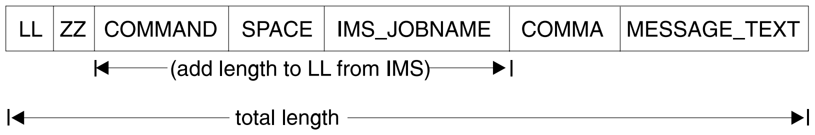
LL	Is a 2-byte field containing the message length (LL + ZZ + MESSAGE_TEXT).

ZZ	Is a 2-byte field containing binary zeroes.
MESSAGE_TEXT	Is the notify message text that IMS expects to receive via the `MODIFY command`. The message text must not be altered.

Issue the `MODIFY` (F) command as follows:

```
MODIFY ims_jobname,message_text
```

The external subsystem must prefix the notify message text passed by IMS with `MODIFY ims_jobname,` (or `F ims_jobname,`) before sending the message. The following shows the format for the SVC 34 command input:



where:

LL	Is a 2-byte field containing the total length of the command input area (COMMAND + SPACE + IMS_JOBNAME + COMMA added to the length in the LL field passed to the Identify exit routine).
ZZ	Is a 2-byte field containing binary zeroes.
COMMAND	Contains the `MODIFY` command verb (C'MODIFY' or C'F').
SPACE	Is a 1-byte field containing a blank (C' ').
IMS_JOBNAME	Is an 8-byte field containing the IMS control region jobname left justified and padded with blanks on the right. The Identify exit routine can obtain the jobname from the TIOT pointed to by the current TCB.
COMMA	Is a 1-byte field containing a comma (C',').
MESSAGE_TEXT	Is the notify message text passed to the Identify exit routine.

Application Program Request Support

Application calls are passed to IMS from the language interface module which must be link-edited with the application program. The language interface branches to the appropriate IMS program request handler passing the application program call parameter list. For calls directed to external subsystems, the language interface must also pass an external subsystem parameter list which it constructs. The purpose of this parameter list is to pass the LIT (language interface token) for the external subsystem to which the call is directed. IMS routes the application call to the external subsystem whose LIT value matches the LIT value passed on the call.

IMS passes both the call parameter list and the external subsystem parameter list to the Normal Call exit routine for the intended external subsystem. The first word of the external subsystem parameter list contains the address of a 4-byte field containing the LIT value in character format, left justified and padded on the right with blanks. IMS prescribes only the first word in the external subsystem parameter list (address of the LIT). The parameter list can be extended to provide external subsystem-dependent information to the Normal Call exit routine.

Language Interface Definition

IMS provides a language interface module, DFSLI000, which supports the value of SYS1. The installation can use this module or it can define its own language interface if it wants to use a LIT value other than SYS1. When two or more external subsystems are accessed by the IMS system, the installation must define its own language interface module(s) because each subsystem has a unique LIT.

IMS provides the DFSLI macro to assist the installation in generating a language interface module. The code necessary to perform the language interface function is generated in the DFSLI macro expansion. The IMS macro library must be supplied when the macro statements are compiled to generate the module.

Related Reading: For more information on using the DFSLI macro, see "Defining the Language Interface Module" on page 553.

Language Interface Entry Points Unique to External Subsystems

The IMS language interface module provides three entry points that application calls directed to an external subsystem can exclusively use. Two of the entry points are associated with an implied LIT value specified with the DFSLI macro. (The language interface module, generated by the DFSLI macro, contains the specified LIT value as a hard-coded constant.) The third entry point is not associated with an implied LIT value; it allows the application program to specify the LIT value when it makes the application call. For all entry points, register 1 contains the address of the parameter list which IMS passes to the external subsystem. The following are language interface entry points to external subsystems:

DSNHLI
Entry point associated with an implied LIT value.

The application program does not need to know which subsystem provides access to the external resources it uses. (If the external subsystem is DB2, this entry point is used for SQL calls.)

DSNWLI
Entry point associated with an implied LIT value.

The application program does not need to know which subsystem provides access to the external resources it uses. (If the external subsystem is DB2, this entry point is used for Instrument Facility calls.)

DFSESS
Entry point allowing an application program to specify an LIT value.

The application program must know which subsystem provides access to the external resources it uses. The application program must specify the address of the LIT value as the first parameter in the application call list. Before it passes control to the external subsystem, IMS increments the address of the application call list by four to skip over the LIT value parameter.

Restriction: This entry point should **not** be used to communicate with a DB2 subsystem.

Accessing Multiple External Subsystems

An application program can access DL/I and an external subsystem in the same execution. Whether or not an application program can access more than one external subsystem in the same execution can be restricted by the language interface.

Where the data (call) interface provided to application programs by one external subsystem (product) is distinct from the interface provided by another external subsystem (for example, DL/I calls vs. SQL calls), an application can access both subsystems because the language interface paths can be different. Where the data interface is the same, as in the case of two external subsystems of the same type (two instances of the external subsystem) or two external subsystem products that use the same call interface (for example, SQL), an application cannot access both in the same execution unless the application is written to be dependent on data location. (The dependency is intrinsic in the case of different call interfaces.)

Resource Recovery Token

A 16-byte recovery token is used to uniquely identify a unit of work across all subsystems to which the application has thread connections. IMS passes the token to the Signon exit routine before the thread is created. For commit and resolve indoubt processing, IMS passes the recovery token to identify the unit of work for which the requested action is to be taken.

The recovery token is constructed as follows:

IMS_id	OASN	commit_number
8	4	4

where:

IMS-id
Is the IMS system id (1 to 4 characters), left justified and padded with blanks on the right to eight bytes.

OASN
Is a 4-byte binary origin application sequence number assigned to the application when it is scheduled. The OASN is assigned based on the scheduling order within the IMS system since the last cold start. It is also referred to as the application schedule number.

commit_number
Is a 4-byte binary commit number. The commit number is initialized to binary zeroes when the application is scheduled and then incremented after each commit is processed for the application.

The external subsystem should check the recovery token passed at signon for uniqueness. Cold starts of IMS can cause a recovery token to be generated that is a duplicate of a recovery token that is indoubt in the external subsystem. The Signon exit routine can indicate to IMS that the recovery token passed was found to be a duplicate in which case IMS terminates the application program with an abend. The Commit Prepare exit routine can also indicate that the token is a duplicate that supports external subsystems that choose not to associate the recovery token with the unit of work until commit is processed.

The installation uses the `/DISPLAY SUBSYS` command with the OASN keyword to determine what units of work are in indoubt status in IMS. The installation can use the `/CHANGE` command (when necessary) to manually delete indoubt units of work in

IMS. The /CHANGE command only affects unit of work status in IMS. There is no communication with the external subsystem. These commands use only the OASN and not the full recovery token; /DISPLAY lists only the OASN portion of the recovery token (in decimal format) and /CHANGE accepts just the OASN (again in decimal format). (Within IMS, the OASN is unique across all known units of work.)

Terminating the External Subsystem Connection

IMS terminates the external subsystem connection (control region connection) in an orderly manner when one of the following occurs:

- IMS processes a /STOP SUBSYS command.
- The external subsystem (ESAP) activates the IMS subsystem termination service exit routine.
- The external subsystem posts the termination ECB provided on the Identify exit routine invocation.
- Certain attach processing errors are encountered.
- IMS is shutting down.

IMS allows existing threads to the external subsystem to complete processing. When all threads have terminated, IMS terminates the connection by activating the Terminate Identify exit routine from the control region.

When the connection is terminated due to a /STOP SUBSYS command, the termination ECB being posted, or processing errors, IMS puts the external subsystem connection in stopped status. Once in stopped status, IMS does not allow a connection to be reestablished. A /START SUBSYS command is required to remove the stopped status.

Termination Requested by the External Subsystem

The external subsystem can cause the connection to be terminated either by posting the termination ECB or by activating the Subsystem Termination Service exit routine from the ESAP. The connection is not put in stopped status when the service is used.

The termination service might be used in conjunction with the external subsystem command exit function. For example, when the command exit routine is activated with an external subsystem termination command supplied on an IMS /SSR command, the exit routine could activate the Subsystem Termination Service exit routine to cause the connection to the external subsystem to be terminated.

Related Reading: The termination service is described in “Subsystem Termination Service Exit Routine” on page 546.

On the initial identify performed in the control region, IMS provides the external subsystem with the address of a termination notification ECB. When the subsystem wishes the connection to be terminated, it should post the ECB. The ECB is located in CSA. Depending on the post code, IMS terminates the connection in the following manner:

> Deactivate all active threads, prohibit the initiation of any new threads, and then terminate the connection. Upon completion of the terminate function, the connection is set in a stopped state.

Supported Post Codes:

Attach Facility

X'40000000'	Reserved for IMS.
X'40000008'	External subsystem is terminating in an orderly fashion.
X'4000xxxx'	All other post codes are interpreted as a quick or catastrophic shutdown of the external subsystem.

Dependent Region Connections

At dependent region termination, the Signoff and Terminate Identify exit routines are **not** activated **unless** the control region connection is to be terminated. That is, it is only when the IMS system will continue to run but without a connection to the external subsystem that the dependent region Signoff and Terminate Identify exit routines are activated. This is the case when either the external subsystem has posted the termination ECB provided on the identify, the ESAP has activated the IMS Subsystem Termination Service, or IMS has processed a `/STOP SUBSYS` command.

When none of these have occurred, IMS does not communicate dependent region termination to the external subsystem. IMS expects the external subsystem to monitor the dependent TCB via an MVS End Of Task (EOT) exit routine. The subsystem should do any signoff and terminate identify processing it requires when the EOT exit routine is notified of the region termination.

The external subsystem must also monitor EOT exit routine for dependent regions for which a thread was created. When an application program terminates abnormally, the Terminate Thread exit routine is not called.

When the control region connection is to be broken, a signoff followed by a terminate identify is done for the dependent region at region termination or after thread termination if a thread was active when the request to break the system connection was received. The Signoff exit routine is called only once even though more than one signon might have been done for the region. IMS continues its signoff and terminate identify processing and does not reactivate these exit routines if they encounter errors.

Explanation of Stopped Status

The installation should be aware of the conditions that cause IMS to stop the external subsystem connection. Once stopped, the `/START SUBSYS` command must be used to reestablish the connection. Stopped status is carried across restarts. The following list includes the conditions that cause the stopped status to be set:

- When the external subsystem posts the subsystem termination ECB provided during identify processing. Regardless of the post code type (for example, orderly or catastrophic), the connection is stopped upon completion of the termination processing.
- On abnormal termination of the IMS task (TCB) in the control region under which the ESAP is activated, the external subsystem connection is marked stopped.

  If the control region abends, it is unlikely the stopped state will be set. After a successful IMS restart, the connection is in the state it was prior to the abend.
- The obvious case is after processing of the `/STOP` command. Even if IMS abends while processing the `/STOP` command, the stopped state is set.
- When IMS restarts, if outstanding recovery tokens exist for an external subsystem that is no longer defined to IMS (for example, the installation deleted its definition from the external subsystem definition member in IMS.PROCLIB), stopped status is set for that external subsystem.

Chapter 77. IMS System Service Exit Routines

The external subsystem can activate IMS-supplied synchronous exit routines to provide prescribed IMS services. External subsystem exit routines can request IMS to:

- Write a log record on the IMS log (Log Service exit routine).
- Send a message to an IMS destination (Message Service exit routine).
- Initiate a connection (Subsystem Startup Service exit routine).
- Terminate a connection (Subsystem Termination Service exit routine).

External subsystem exit routines are organized alphabetically and described on “Log Service Exit Routine” on page 542.

General System Service Exit Routine Interface

To activate an IMS system service exit routine, the ESAP loads the address of an IMS service router module from EEVPESGL in the EEVTP control block and branches to it. The ESAP passes required parameters via an exit routine parameter list (EPL) in the same general format as the EPL passed to external subsystem exit routines. Each exit routine has a unique function code. The address of the function code defined for the exit routine being activated must be supplied in the EPL. If a function code address is not passed, the invalid parameter list return code, X'20', is returned to the caller.

Upon entry, the system service exit routine saves all registers using the provided save area. The registers contain the following:

Register	**Contents**
1	Address of exit parameter list (EPL)
13	Address of save area
14	Return address
15	Address obtained from EEVPSEGL in the EEVTP

Before returning to the ESAP, the system service exit routine restores all registers except register 15, which contains a return code. The parameters and return codes for each system service exit routine are described in the sections following.

The storage key of the save area should be that of the caller, for example, key 8 for the Normal Call exit routine; key 7 for the Subsystem Termination exit routine.

The key of the storage passed to the system service exit routines (Log exit routine, Message exit routine) must be the same as the caller’s. For example, if the Identify exit routine wants to place data on the IMS log, that data must exist in storage obtained while the Identify exit routine was running in key 7 before calling the Log exit routine.

IMS system service exit routines should not be activated from TCBs attached by the ESAP. The system service modules expect to be activated under an IMS internal structure that is only available under IMS TCBs.

Log Service Exit Routine

An external subsystem uses this exit routine when it wants IMS to write a log record. IMS reserves log record type X'55' for external subsystem usage. The exit routine does not accept any other log record types.

Activating the Routine

The EPL contains:

Offset		
Hex	**Decimal**	**Content**
0	0	Address of the parameter count field. The count field must contain the value F'3'.
4	4	Address of the 1-character logging service function code, X'16'. If an address is not present, return code X'20' is returned to the caller.
8	8	Address of the EEVT prefix. The EEVT prefix must be fully initialized.
C	12	Address of the log record area. The exact contents are written to the log. IMS does not alter this area. The log record must be in the following format:

|◄—log record prefix—►|

LL	ZZ	C	T	RC	SSN	Data
2	2	1	1	2	8	variable

The log record prefix must contain the LL, ZZ, and C fields as follows:

LL A 2-byte field that must contain the total length of the record. The total length (prefix + data) cannot exceed the logical record length (LRECL) for the system log data set minus 4 bytes.

ZZ A 2-byte field that must contain binary zeroes.

C A 1-byte field that must contain the log record type, X'55'.

Recommendation: The remaining prefix fields are suggested to allow log records for a particular external subsystem to be identified.

IMS does not check the contents of the fields.

T A 1-byte field that contains a system or function type.

RC A 2-byte field that contains a reason code.

SSN An 8-byte field that contains the external subsystem name.

Contents of Register 15 on Return

Return Code	Meaning
0	Log request successful. The log record passed by the caller was written to the IMS log.
4	Log request unsuccessful. A failure (or error) was encountered in attempting to process the log request. The record was not logged.
8	Log request unsuccessful. The caller attempted to log an invalid log record type. Only type X'55' log records are accepted by the Log exit routine. The record was not logged.

Return Code	Meaning
20	Log request unsuccessful. Invalid data was detected in the exit parameter list. The record was not logged. Possible errors: • Invalid log record address—the EPL contains zeroes or a negative value. • Invalid log record length—the length field contains zeroes or a negative value.

Message Service Exit Routine

An external subsystem uses this exit routine when it wants IMS to send a message to an IMS destination (MTO or input terminal). Two types of messages are accepted by the exit routine: preedit (prebuilt) and key call (message number).

Preedit messages are assumed to be formatted by the caller. Single or multiple segment preedit messages are accepted. The address of the message is passed to the exit routine.

On a key call, the address of the message number is passed. The message itself is supplied in the user message table.

Related Reading: For more information, refer to “Chapter 28. User Message Table (DFSCMTU0)” on page 213.

Activating the Routine

The EPL contains:

Offset Hex	Decimal	Content
0	0	Address of the parameter count field. The count field must contain the value F'5'.
4	4	Address of the 1-character message service function code, X'15'. If an address is not present, return code X'20' is returned to the caller.
8	8	Address of the EEVT prefix. The EEVT prefix must be fully initialized.
C	12	Address of the message area for preedit messages; address of the message number for key call (user) messages. IMS assumes the message number to be a two-byte binary value. IMS does not alter the message area.
10	16	Address of the 8-byte alphanumeric destination name (that is, the LTERM name). If this field contains binary zeroes, the default message destination is the IMS master terminal. The name is left justified and padded with blanks on the right.

Offset Hex	Decimal	Content
14	16	Address of the 2-character message type indicators. The first character indicates the message type. The second character indicates system versus pageable message.

The supported message types in character format are:

S Message is a single segment and can be preedited.

M Message might consist of a variable number of segments, each segment is prefixed by its own length. The entire message is prefixed by the entire message length.

U Indicates a user message is sent. When specified, IMS expects the address of a message number passed in the input parameter list. The message is assumed to be present in the IMS user message table.

The second character indicates to IMS where the message is to be placed on the master terminal in the event the MVS split screen master terminal format is used. Traditionally, the top part of the screen is reserved for unsolicited output such as I/O error messages and immediate command response message (that is, command complete, command in progress, invalid keyword or parameter).

The lower portion of the screen is where display output (`/DISPLAY LTERM ALL`) is directed. This allows the MTO to page through the output message using the PA keys. If the split screen master terminal format is not being used, output is displayed, taking advantage of the entire screen.

The supported screen format indicators in character format are:

S Indicates the message being sent is to be placed on the top portion of the screen (system area)

P Indicates the message being sent is to be placed on the lower portion of the screen (pageable area, typically used for `/DISPLAY` command output)

The format for single segment messages (message type **S**) is:

```
  OFFSET          LENGTH/
 HEX  DEC  NAME   ALIGNMENT      DESCRIPTION

   0    0  MSGLL     2       record length
   2    2  MSGZZ     2       reserved bytes
   4    4  MSGDATA   nnn     substance of message
```

The format for multiple segment messages (message type **M**) is:

```
  OFFSET          LENGTH/
 HEX  DEC  NAME   ALIGNMENT      DESCRIPTION

   0    0  MSGLL     2       total record length
   2    2  MSGZZ     2       reserved bytes
   4    4  MSGLL     2       message segment length
   6    6  MSGZZ     2       reserved bytes
   8    8  MSGDATA   10      first segment of message
  12   18  MSGLL     2       message segment length
  14   20  MSGZZ     2       reserved bytes
  16   22  MSGDATA   20      second segment of message
```

Contents of Register 15 on Return

Return Code	Meaning
0	Message request successful. The message was enqueued to the master terminal or the specified destination.
4	Message request unsuccessful. The message did not pass validity checking. The message was not sent.
8	Message request unsuccessful. A failure occurred in processing the message request. The message was not sent.
20	Message request unsuccessful. Invalid data was detected in the exit parameter list (that is, invalid destination name specified). The message was not sent. Possible errors: • Invalid EEVTP address—the EPL contains zeroes. • Invalid destination name—the destination name area contains blanks. • Invalid message type.

Subsystem Startup Service Exit Routine

The IMS Subsystem Startup Service exit routine is activated by an external subsystem exit routine to cause IMS to initiate a connection to the external subsystem. If the control region identify or an MPP or IFP dependent region identify was deferred (for example, not done automatically during region initialization processing), this service is to be used to establish the necessary connection. It can also be used to establish the connection for a BMP dependent region. The external subsystem activates the service from the Subsystem Not Operational exit routine in the dependent region (key 8) when the first application call to the external subsystem is processed in the dependent region.

Related Reading: See “Initiating the External Subsystem Connection” on page 533 for further discussion.

If the control region connection has not been established, the Startup Service exit routine activates the control region identify process before activating the dependent region identify process. The startup exit routine activation fails if the notify message passed to the control region Identify exit routine (on a previous activation) was accepted by the exit routine but the external subsystem has not sent the message to IMS to indicate that it is ready to establish a connection.

Activating the Routine

The EPL contains:

Offset Hex	Decimal	Content
0	0	Address of the parameter count field. The count field must contain the value F'2'.
4	4	Address of the one-character startup service function code, X'17'. If an address is not present, return code X'20' is returned to the caller.
8	8	Address of the EEVT prefix. The EEVT prefix must be fully initialized.

Contents of Register 15 on Return

Return Code	Meaning
0	Subsystem connection successful. The identify has been performed. IMS continues normal processing.
4	Subsystem connection unsuccessful. The identify process initiated by IMS resulted in the notify message being queued to the external subsystem. However, the subsystem has not yet been started. IMS waits for the receipt of the notify message before continuing the connection process.
8	Subsystem connection unsuccessful. The control region identify was previously attempted, at which time the Identify exit routine accepted the notify message. IMS continues to wait for the external subsystem to send the notify message before reactivating the exit routine to establish the control region connection.
C	Subsystem connection unsuccessful. The connection attempt failed either in the ESAP or in the external subsystem.
10	Subsystem identify unsuccessful. IMS is shutting down.
14	Subsystem connection unsuccessful. IMS is notified that the external subsystem is terminating either in a quiesce or catastrophic fashion.
18	Subsystem connection unsuccessful. During the connection initialization process, the external subsystem Initialization exit routine was activated. The exit routine responded with the `never connect` return code (return code 8).
1C	Subsystem connection unsuccessful. Resources required to process the connection request were unavailable. When this condition exists, IMS sends message DFS3620I to the MTO indicating the resource type. When a required exit routine is missing, message DFS3608 is also sent to the MTO indicating the exit routine name. Refer to *IMS/ESA Messages and Codes*for more information on these messages.
20	Subsystem connection unsuccessful. Invalid data was detected in the exit parameter list. IMS did not initiate the startup (identify) process.
24	Subsystem connection unsuccessful. The startup request is rejected because an external subsystem exit activated the IMS Terminate Service exit routine. IMS is still in the process of terminating the connection between the two subsystems. Additional dependent region connections are prohibited while in this state.
28	Subsystem connection unsuccessful. The startup request is rejected because IMS is terminating the connection due to a `/STOP SUBSYS` command entered. Additional dependent region connections are prohibited while in this state.

Subsystem Termination Service Exit Routine

This IMS-provided exit routine prohibits new connections to the external subsystem but allows the existing connections to terminate normally (quiesce). When all dependent region connections are terminated, the control region connection is terminated.

The subsystem Termination exit routine is activated by the ESAP, possibly when a subsystem termination command is intercepted by the ESAP exit routine. The intercepting routine is available whether or not the subsystem is attached and running in user key.

To perform the necessary processing, the subsystem Termination exit routine switches to key 7 supervisor state.

Activating the Routine

The EPL contains:

Hex	Decimal	Content
0	0	Address of the parameter count field. The count field must contain the value F'2'.
4	4	Address of the one-character termination service function code, X'18'. If an address is not present, return code X'20' is returned to the caller.
8	8	Address of the EEVT prefix. The EEVT prefix must be fully initialized.

Contents of Register 15 on Return

Return Code	Meaning
0	Subsystem termination successful. IMS terminates subsystem connections in all dependent regions. When the dependent region connections are all terminated, the control region connection is terminated.
4	Subsystem termination unsuccessful. A failure (or error) was encountered in attempting to process the termination request. IMS terminates the connection to the subsystem as stated under return code 0.
20	Subsystem termination unsuccessful. Invalid data was detected in the exit parameter list. IMS did not initiate termination processing.

Chapter 78. Installing External Subsystems with IMS

This chapter describes installation tasks required for accessing external subsystems. There are tasks that you need to do for your external sections:

- Installation tasks for your IMS subsystem:
  1. Define your external subsystems to IMS:
     - Add a member to the IMS procedure library (IMS.PROCLIB) that contains the information about each external subsystem with which IMS communicates.
     - On the execute procedures for the control region, message processing region, Fast Path region, or batch message processing region, specify the execution parameter SSM=, such that the concatenation of the IMS system ID with SSM is the name of the IMS.PROCLIB member that contains the external subsystem definition.
  2. Define a language interface module, if you want to use one other than that supplied by IMS.
  3. Specify on the IMS `OPTIONS` statement whether or not tracing of activity on the IMS-external subsystem link will occur.
- Installation tasks for your external subsystems:
  1. Provide the following:
     - External subsystem module table (ESMT).
     - Resource translation table (RTT) (optional).
  2. Ensure that the external subsystem modules and tables used by IMS are in an appropriate APF-authorized library.

Installation Tasks for the IMS User

This section describes the installation tasks the IMS user needs to perform.

Defining Your External Subsystems to IMS

In order for IMS to initiate contact with an external subsystem, it is necessary to add a member to the IMS procedure library, IMS.PROCLIB. This member contains an entry for each external subsystem with which IMS communicates.

Using the IMS.PROCLIB data set allocated prior to Stage 2 of IMS system definition and the MVS utility IEBUPDTE, create the PROCLIB member(s). The first 1 to 4 characters of this PROCLIB member's name must match the IMSID parameter specified on the IMSCTRL macro statement or overridden using the execution parameter IMSID=. The last 4 positions represent any unique installation-defined identifier. This identifier is appended to the IMSID to construct the member name.

Related Reading:
- For more information on the MVS utility IEBUPDTE, see *DFSMS/MVS V1R1 Utilities*.
- For more information on the SSM= parameter, see *IMS/ESA Installation Volume 2: System Definition and Tailoring*.

Each entry within the PROCLIB member consists of a blocked or unblocked 80-character record. Information for each record begins in position 1 of the record.

Specifying Member Entry Parameters

Some of the keyword parameters are required and are applicable for any external subsystem; other parameters are optional and can be unique to the external subsystem type. The SSM PROCLIB member entry parameters are listed in “SSM PROCLIB Member Entry Keyword Parameters”. Each parameter description includes information on whether the parameter is required or optional.

The following are guidelines for specifying member entries:

- You can specify certain parameter values with alphanumeric characters (beginning with A-Z, $, #, or @, and containing alphabetic characters or the digits 0-9).
- Specifying duplicate subsystem names within a member is invalid. IMS uses the first definition for a specific subsystem name.
- IMS issues DFS3600I messages if it encounters any errors during SSM PROCLIB member processing. (For more information, see the *IMS/ESA Messages and Codes*.)
- Use a comma to delimit keywords parameters. You can code the parameters in free format with leading and trailing blanks.
- You can have keyword format statements be continued by placing a non-blank character in position 72. The last character prior to position 72 must be a comma (to delimit the preceding parameter). Continue a statement starting in positions 2-72; a continued statement cannot start in position 1.
- You can add comments by placing a ’*’ in position 1.

SSM PROCLIB Member Entry Keyword Parameters

Use the following keyword parameters to define your external subsystems to IMS. The SST= parameter must begin in position 1, signaling that keyword format is being used.

SST=

Is a **required** one- to eight-character, left-justified name of the *subsystem type* to which IMS connects. The SST= parameter identifies an external subsystem and can be used to identify processing that is unique to that type of external subsystem. The SST= parameter also indicates the start of a new subsystem definition and must begin in position 1. This parameter is **required** for all PROCLIB member entries defined in keyword format.

For information on the required and optional parameters for specific external subsystems, see “Parameters for Specific External Subsystems” on page 552.

SSN=

Is a **required** one- to four-character alphanumeric MVS *subsystem name* of the external subsystem that will be attached. This name is specified by your installation, and must be identified to MVS during MVS system generation.

The external subsystem does **not** have to be an MVS subsystem. However, if it is **not** a subsystem, some external subsystem processing functions cannot be used, such as Subsystem Interface (SSI) and End of Task/End of Memory (EOT/EOM) processing.

LIT=

Is a **required** one- to four-character alphanumeric field specifying the *language interface token*. The language interface provides routing information to IMS and also contains external subsystem-dependent information for the IMS applications. IMS provides a language interface module named DFSLI000, which uses a token value of `SYS1`. To use this language interface, the token value SYS1 is specified for the LIT parameter to relate this language interface to an external subsystem. If you do not want to use this supplied language interface module, or want to use more than one (for example, to access test and production subsystems), assemble and link your user-defined language interface, with a token value other than SYS1, into an installation-selected library.

This token must then be specified on a record in conjunction with or instead of the standard language interface token, SYS1. Each IMS control or dependent region can thus access alternative external subsystems via the corresponding language interface module. Each individual application program, however, can be link-edited to only one language interface module, and can thus access only one external subsystem. Each application programmer must include the appropriate language interface module during the link-edit process for each application program that calls the external subsystem from an MPP, IFP, or BMP dependent region.

For information on link-editing the language interface module into an appropriate IMS library or defining your own language interface module, see “Defining the Language Interface Module” on page 553.

ESMT=
Is a **required** one- to eight-character alphanumeric parameter specifying the name of the *external subsystem module table*. This table is defined in the external subsystem and specifies the external subsystem modules IMS must load, and the work areas that IMS must create.

It is the responsibility of the external subsystem to create the external subsystem module table and to make the name of that table known to the IMS user who will define these PROCLIB member entries.

RTT=
Is an **optional** one- to eight-character alphanumeric parameter specifying the name of the *resource translation table*. This table maps an application program name to the name used by the external subsystem for the entity that defines the resources needed by the application (if these names are not the same). The resource translation table need only be used if IMS and the external subsystem use different names for the same resource.

It is the responsibility of the external subsystem to create the resource translation table and to make the name of that table known to the IMS user who defines these PROCLIB member entries.

REO=
Is an **optional** one-character *region error option* code. The option specified here determines the action IMS takes when an application program issues a request for external subsystem services before connection to the external subsystem is complete or if problems are encountered with the external subsystem. IMS does not use these values, but passes them to the external subsystem for its use. The possible values for this command are:

R The application program receives a return code indicating that the request for external subsystem services has failed. If no other value is specified for the region error option, R is the default.

Q IMS abnormally terminates the application program with an abend code of U3051. IMS then places the transaction on a queue to be processed when the connection to the external subsystem is established. After a transaction is on this queue, it can be recovered across warm starts and emergency restarts of IMS.

A IMS abnormally terminates the application program with an abend code of U3047 and discards the transaction input.

CRC=
Is an **optional** one-character command recognition character. Any EBCDIC value except / is permitted (/ is reserved for IMS). This parameter is specified if you want to allow external subsystem commands to be entered from IMS terminals or automated operator interface (AOI) applications. Commands to be executed in the external subsystem are entered at an authorized IMS terminal or by an AOI application by first entering `/SSR`, followed by the command recognition character specified here, followed by the external subsystem command. When IMS receives a command in this format, it routes that command to the external subsystem for processing.

Related Reading: For additional information on the command, see the description of `/SSR` in *IMS/ESA Operator's Reference*.

The external subsystem can require IMS user IDs and LTERM names to be authorized to issue external subsystem commands. Refer to the external subsystem documentation for information on command authorization requirements.

SSM PROCLIB Member Example

SST=DB2,**SSN=***ssname*,**ESMT=***esmtname*

Parameters for Specific External Subsystems

The following are required and optional parameters for specific subsystem types.

DB2 External Subsystems: Required parameters:

```
SST=DB2,SSN=,LIT=,ESMT=
```

Optional parameters:

```
RTT=,REO=,CRC=
```

Using the SSM PROCLIB Member

Once you've defined a PROCLIB member, you must do the following to use the member:

1. Insert a DD statement for IMS.PROCLIB with the ddname PROCLIB in the step execution JCL.
2. Specify the correct PROCLIB member on the EXEC statement parameter SSM= or on the `/START SUBSYS SSM` command.

Specifying the SSM= EXEC Parameter

The SSM= EXEC parameter is required to enable your installation to indicate which PROCLIB member contains an entry for the external subsystem you want to attach. You can specify the SSM= EXEC parameter on the EXEC procedures of that control region, or for an MPP, IFP, or BMP dependent region.

If the SSM= EXEC parameter is not specified at the time the IMS subsystem is initialized, the ESAF component is not activated. In this case, you can shutdown IMS and restart it specifying the SSM= EXEC parameter to activate the ESAF component and allow external subsystem support.

If the ESAF component is not activated and a connection to an external subsystem is required, the `/START SUBSYS` command can be entered by specifying the SSM keyword. This action activates the ESAF component and attempts a connection to the external subsystems defined in the SSM PROCLIB member.

Related Reading: For more information on the command, see *IMS/ESA Operator's Reference*.

For an MPP or an IFP dependent region that is currently active but not connected, you must stop and start the region for a connection for that dependent region.

When SSM= is specified for the control region, any dependent region running under the control region can attach to the external subsystems named in the PROCLIB member pointed to by the SSM= parameter.

When SSM= is specified for a dependent region, it provides a filter that determines whether a given application program running in that dependent region can access external subsystem resources. For example, if a subsystem entry for a given external subsystem exists in the named dependent region member, but not in the control region member, a message is sent to the MTO during dependent region initialization and a subsystem connection is not made for this dependent region entry.

If SSM is specified on the control region but the installation wants a particular dependent region to be prohibited from attaching to any external subsystem, a PROCLIB member must be created having **no** entries, and its ID specified in the dependent region. Application programs in this dependent region cannot attach to any external subsystem.

When the PROCLIB member contains a null record, you'll receive the following informational message:

```
DFS3600I UNABLE TO INITIATE THE EXTERNAL SUBSYSTEM TABLE, RC=36
```

Related Reading: For an explanation of this message, see *IMS/ESA Messages and Codes*.

If an invalid member name is specified or if the record format is incorrect, no external subsystem is attached, but initialization continues.

Supported Positional Parameters

An SSM PROCLIB member can contain both keyword format statements and positional format statements in any order. However, you cannot mix keyword and positional parameters in a single subsystem definition. The following positional parameters are accepted:

```
SSN,LIT,ESMT,RTT,REO,CRC
```

Recommendation: It is recommended that you use keyword parameters are described in "Specifying Member Entry Parameters" on page 550.

Defining the Language Interface Module

In order for an IMS application program to use the language interface module, the application program modules must be link-edited using the IMS reenterable DL/I language interface. The true attributes must be specified; for example, if a module is reusable only, don't specify "RENT".

Restriction:IMS does not support dynamic calls to the language interface.

If the installation wants to make dynamic calls, it needs to provide the RMODE value at link-edit time so that the language interface will be shown below the 16MB line.

The IMS/360 DL/I language interface is not reenterable. Any IMS/360 application programs that were designed to be reenterable or serially reusable can use external subsystem resources only after being link-edited with the IMS language interface.

IMS provides a language interface module, DFSLI000, which supports the external subsystem interface. DFSLI000 passes a language interface token value of SYS1. The installation can use this module or it can define its own language interface if it wants to use a LIT value other than SYS1. When two or more external subsystems are accessed by the IMS system, the installation must define its own language interface module(s) because each subsystem has a unique LIT.

IMS provides the DFSLI macro to assist the installation in generating a language interface module. The code necessary to perform the language interface function is generated in the DFSLI macro expansion. The IMS macro library must be supplied when the macro statements are compiled to generate the module. The format of this macro is as follows, where user-supplied values must be substituted for the x's.

```
DFSLIxxx   DFSLI   TYPE=V2DB,LIT=xxxx
           END
```

where:

DFSLI*xxx*
Specifies the CSECT name for the module.

Recommendation: It is recommended that the CSECT name match the link-edit name of the language interface module.

For the IMS-supplied language interface module, the value for this parameter is DFSLI000.

TYPE
Specifies the language interface type to IMS. The only value that you can enter for this parameter is V2DB.

LIT
Specifies the language interface token that relates this language interface to an external subsystem by means of their respective specifications in a PROCLIB member entry as the LIT and SSN parameters. The IMS-supplied language interface module DFSLI000 uses a LIT value of SYS1.

When IMS issues an external subsystem request, IMS knows the target subsystem by the LIT used in the request. For example, consider the case of an IMS dependent region accessing two external subsystems ESS1 and ESS2.

- You already have a default language interface module DFSLI000, as specified by the default LIT, SYS1. You now generate a second language interface DFSLI001 with a LIT of SYS2.
- Define two entries in the PROCLIB SSM member. The first entry points to ESS1 with LIT=SYS1. The second entry points to ESS2 with LIT=SYS2.
- Link-edit IMS application programs accessing ESS1 with the IMS-provided language interface module DFSLI000.

- Link-edit IMS application programs accessing ESS2 with the language interface module that you have generated, DFSLI001.

Although a region can communicate with two or more external subsystems, an IMS application can access only the external subsystem referred to in the link-edited language interface. However, you can alter the SSM member to route application requests to a different external subsystem from the subsystem defined in the language interface. Alternatively, you can use the DFSESS entry point in the language interface module.

Related Reading: For more information, see “Language Interface Entry Points Unique to External Subsystems” on page 537.

Specifying the Desired Trace Options

If you want to trace the connection and disconnection activities of IMS to the external subsystem, the parameter SUBS= on the OPTIONS statement must be set to ON. Specifying OUT on this parameter causes trace information to be written to the log. If this parameter is omitted, traces can be turned on and off by means of the `/TRACE` command.

Installation Tasks for the External Subsystem User

This section describes the tasks that the External subsystem user needs to perform.

Providing Appropriate Tables in the External Subsystem

It is the responsibility of the external subsystem user to provide the external subsystem module table and, optionally, the resource translation table in the external subsystem. It is also the user’s responsibility to make the names of these tables known to the person who defines the IMS.PROCLIB member entries. The person defining the PROCLIB entries includes the names of these tables as values for the parameters ESMT and RTT, respectively.

Placing External Subsystem Modules and Tables in Appropriate Library

IMS loads the modules and tables specified in the external subsystem module table from the external subsystem libraries when the SSM= parameter is specified in an IMS region. To make this process successful, make the external subsystem libraries (which must be APF authorized) available to IMS. To do this, you can either add the external subsystem libraries to the `JOBLIB/STEPLIB/LINKLIST` concatenation, or you can create a `DFSESL DD` statement for this purpose

IMS does not automatically generate a `DFSESL DD` statement).

If `JOBLIB/STEPLIB/LINKLIST` concatenation is not authorized, you must use the DFSESL DD statement. For online IMS regions, the subsystem library or libraries must be concatenated after the library containing the IMS modules (usually IMS RESLIB). When multiple subsystems are connected, additional subsystem data sets can be concatenated.

```
//DFSESL DD  DISP=SHR,DSN=IMS.RESLIB
//       DD  DISP=SHR,DSN=DSNxxx.DSNLOAD
//       DD  DISP=SHR,DSN=DSNyyy.DSNLOAD
```

The IMS.RESLIB is first, followed by the subsystem libraries.

Part 9. Appendixes

Appendix A. DBCTL Function Requests

This appendix documents General-Use Programming Interface and Associated Guidance Information. See “Notices” on page xxi to understand this classification of information.

DBCTL allows access to IMS full-function databases and to data entry databases (DEDBs) from a coordinator controller (CCTL). The Database Resource Adapter (DRA) is the interface between DBCTL and a CCTL.

DRA Function Requests

This section discusses the requests available to the CCTL that allow it to communicate with DBCTL. These requests are passed to the DRA via the PAPL.

For all DRA requests, there are PAPL fields that the CCTL must fill in. When the DRA completes the request, there are some output PAPL fields that the DRA fills in. Some fields in the returned PAPL might contain the original input value.

(The PAPLTTOK and PAPLUSER fields **do** retain the original input values.)

The PAPLUSER field is a field to be used at the CCTL’s discretion. One possible use for it is to pass data to exit routines.

The DRA returns a code (in the PAPLRETC field) to the CCTL after processing a DRA request. The code indicates the status of the request and can be either an IMS code, a DRA code, or an MVS code. Failed DRA requests return a nonzero value in the PAPLRETC field.

Related Reading:
- For more information on the codes returned when a DRA request fails, see *IMS/ESA Application Programming: Database Manager*.
- For a complete list and description of all DRA return codes, see *IMS/ESA Messages and Codes* and *MVS/ESA System Codes*.

INIT Request

The INIT request initializes the DRA. The DRA startup parameter table contains all of the required parameters that you need to define the DRA. You can use the parameters given in the default module, DFSPZP00, or you can write your own module and link-edit it into the IMS.RESLIB.

Related Reading: For more information on enabling the DRA, see *IMS/ESA Application Programming: Database Manager*.

The INIT PAPL also contains some parameters needed to initialize the DRA. If the same parameter appears in both the INIT PAPL and in the DRA startup parameter table, the specification in the INIT PAPL will override that in the startup table.

In addition to the required parameters, you can also include the following optional parameters in the INIT PAPL:

Field	Contents
PAPLFUNC	PAPLINIT

Field	Contents
PAPLSUSP	The address of the Suspend exit routine
PAPLRESM	The address of the Resume exit routine
PAPLCNTL	The address of the Control exit routine
PAPLTSTX	The address of the Status exit routine

After the INIT request and the startup table have been processed, the DRA returns the following data to the CCTL in the INIT PAPL:

Field	Description
PAPLDBCT	The DBCTL identifier (this is the IMSID parameter generated from the DBCTL startup).
PAPLCTOK	The request token that identifies the CCTL to the DRA.
PAPLTIMO	DRA TERM request timeout value (in seconds).
PAPLRETC	A code returned to the CCTL specifying the status of the request.
PAPLDLEV	A flag indicating to CCTL which functions the DRA supports. (For the latest version of PAPL mapping format see the IMS.GENLIBB library; member name is DFSPAPL.)

INIT Request, Identify to DBCTL

To make the DRA functional, the DRA must identify itself to DBCTL, thus establishing a link between DBCTL and the CCTL. The identify process occurs in two cases:

- As a direct result of an INIT request.
- As part of a terminate/reidentify request from a Control exit routine invocation. (For additional information on the Control exit routine, see *IMS/ESA Application Programming: Database Manager*.)

The DRA identifies itself to the DBCTL subsystem specified in the final DRA startup parameters. The identify process executes asynchronously to the INIT process. Therefore, it is possible for the INIT request to complete successfully while the identify process fails. In this case, the Control exit routine notifies the CCTL that the connection to DBCTL failed.

Related Reading: For more information on responses from the Control exit routine, see *IMS/ESA Application Programming: Database Manager*.

If DBCTL is not active, the console operator will receive a DFS690 message (a code of 0 was returned in the PAPLRETC field). You must reply with either a CANCEL or WAIT response. If you reply with WAIT, the DRA waits for a specified time interval before attempting to identify again. The waiting period is necessary because the identify process won't succeed until the DBCTL restart process is complete. You specify the length of the waiting period on the TIMER DRA startup parameter. If subsequent attempts to identify fail, the console operator will receive message DFS691, `WAITING FOR DBCTL`.

If the DRA cannot identify to DBCTL because the subsystem does not reach a restart complete state, there are two ways to terminate the identify process:

- The Control exit routine is called with each identify failure. This sets a PAPL return code of 4 or 8, which terminates the identify process.
- The CCTL can issue a TERM request.

If you reply with CANCEL to message DFS690, control is passed to the Control exit routine, and the DRA acts upon the routine's decision.

After the identify process successfully completes, the DRA makes the CCTL address space non-swappable and calls the Control exit routine with a list of in-doubt UORs. If no in-doubt UORs exist, a null list is passed. The CCTL can use the RESYNC request to resolve any in-doubt UORs that do exist.

The INIT request will attempt to create the MINTHRD number of thread TCBs. The actual number of TCBs created might be less than this value due to storage constraints.

INIT Request after a Previous DRA Session Termination

If a prior DRA session ended with a TERM request that received a PAPL return code=0, this INIT request must specify PAPLCTOK=0. If PAPLCTOK other than 0 is sent, the INIT request will fail.

The INIT request must pass the prior session's PAPLCTOK value in the current PAPLCTOK field if a DRA session ended one of the following ways:

- A nonzero return code from a TERM request
- An internal TERM request from a Control exit routine request
- A DRA failure

RESYNC Request

The RESYNC request tells DBCTL what to do with in-doubt UORs. The following four subfunction values indicate possible actions:

Value	**Action**
PAPLRCOM	Commit the in-doubt UOR.
PAPLRABT	Abort the in-doubt UOR. Changes made to any recoverable resource are backed out.
PAPLSCLD	The UOR was lost to the transaction manager due to a coldstart.
PAPLSUNK	The in-doubt UOR is unknown to the CCTL. This can occur when the CCTL's in-doubt period does not include the start of phase 1. (See information on two-phase sync point processing in *IMS/ESA Application Programming: Database Manager* for an illustration of in-doubt periods.)

You must fill in the following input fields of the PAPL:

Field	**Contents**
PAPLCTOK	Request token. This token identifies the CCTL to the DRA. The DRA establishes the token and returns it to the CCTL in the parameter list on the startup INIT request. The request token must be passed on to the DRA for all RESYNC requests.
PAPLRTOK	Recovery token. This 16-byte token is associated with a UOR. The first 8 bytes must be the transaction manager subsystem ID. The second 8 bytes must be unique for one CCTL thread. This is one of the in-doubt recovery tokens passed to the Control exit routine.
PAPLFUNC	PAPLRSYN.
PAPLSNC	One of the four values listed above.

TERM Request

The TERM request results in a termination of the DBCTL/CCTL connection and a removal of the DRA from the CCTL environment. The DRA terminates after all threads have been resolved. No new DRA or thread requests are allowed, and current requests in progress must complete.

You must fill in the following input fields in the PAPL:

Field	Contents
PAPLFUNC	PAPLTERM, DRA terminate function code
PAPLCTOK	The DRA request token (output from an INIT request)

After receiving the TERM request results, the CCTL may remove DFSPPRC0.

The following output fields are returned in the PAPL to the CCTL:

Field	Contents
PAPLRETC	The return code
PAPLMXNB	The number of times the maximum thread count was encountered during this DRA session
PAPLMTNB	The number of times the minimum thread count was encountered during this DRA session
PAPLHITH	The largest number of thread TCBs that were scheduled during this DRA session
PAPLTIMX	The elapsed time at maximum thread for this DRA session

Thread Function Requests

The Thread Function requests consist of the SCHED, IMS, SYNTERM, PREP, COMTERM, ABTTERM, and TERMTHRD requests and are described in this section.

SCHED Request

The SCHED request schedules a PSB in DBCTL. The first SCHED request made by a CCTL thread requires a new DRA thread. If any existing DRA thread TCBs are not currently processing a DRA thread, one of these is used. If no TCBs are available, the DRA either creates a new thread TCB (until the maximum number of threads as specified by the MAXTHRD parameter in the INIT request is reached), or makes the SCHED request wait until a thread becomes available.

The value in the PAPLWCMD field indicates whether the thread to which the SCHED request applies is a short or long thread. The type of thread determines the action that IMS takes when a database command is entered for a database scheduled to the thread. The `/STOP DATABASE`, `/DBDUMP DATABASE`, or `/DBRECOVERY DATABASE` command issued against a database scheduled on a short thread will wait for the database to be unscheduled. IMS rejects these commands if they are entered for a database scheduled on a long thread.

You must fill in the following input fields in the PAPL:

Field	Contents
PAPLFUNC	PAPLTFUN, thread function code.
PAPLSFNC	PAPLSCHE, schedule request subfunction code.
PAPLCTOK	The DRA request token (output from an INIT request).
PAPLTTOK	The thread token set up by the CCTL.

Field	Contents
PAPLRTOK	The 16-byte UOR token (RTOKEN). For more information about UORs, see information on sync points in *IMS/ESA Application Programming: Database Manager*.
PAPLPSB	The PSB name.
PAPLWRTH	Deadlock Worth Value. If this thread hits a deadlock condition with any other DRA thread or with any IMS region, DBCTL collapses the thread with the lower deadlock worth value.
PAPLWCMD	This bit defines the thread as either a short or long thread which determines what action IMS takes on a `/STOP DATABASE`, `/DBDUMP DATABASE`, or `/DBRECOVERY DATABASE` command for a database scheduled to the thread. If the bit is set on (X'80'), the database is scheduled on a short thread; if the bit is set off, the database is scheduled for a long thread.
PAPLFTRD	Fast Path Trace Option. If this bit is on (X'40'), Fast Path tracing in DBCTL is activated.
PAPLKEYP	Public Key Option. If this bit is set (X'10'), DBCTL will build UPSTOR area in a special subpool so that applications running in public key can fetch the UPSTOR area.
PAPLLKGV	Lockmax Option. If this bit is set (X'08'), DBCTL uses the value in PAPLLKMX as the maximum number of locks that this UOR can hold. Exceeding the maximum results in a U3301 abend.
PAPLLKMX	Lockmax Value, 0 to 255. This value overrides any LOCKMAX parameter specified on the PSBGEN for the PSB referenced in the SCHED request.
PAPLALAN	Application language type.

Specifying the following input field is optional:

Field	Contents
PAPLSTAT	Address of an area where scheduled statistical data is returned to the CCTL.
PAPLPBTK	Address of the token for the MVS Workload Manager performance block obtained by the CCTL. You must specify this field for MVS Workload Manager support for DRA threads.

The following output fields are returned in the PAPL to the CCTL:

Field	Contents
PAPLRETC	The return code.
PAPLCTK2	The thread request token number 2. This is another DRA token required on future DRA requests originating from this thread.
PAPLPCBL	The address of the PCB list. There is one entry in the list for each PCB in the PSB that was scheduled, even if the PCB cannot be used with DBCTL.
PAPL1PCB	The address of the PCBLIST entry pointing to the first database PCB.

PAPLIOSZ	The size of the maximum I/O area.
PAPLPLAN	The language type of the PSB.
PAPLMKEY	The maximum key length.
PAPLSTAT	The address of the schedule statistical data area. This address must be specified on the input field.

CCTLs currently using the IMS Database Manager and migrating to DBCTL will experience a change in the PCBLIST and user PCB area on a schedule request. The first PCB pointer in the PCBLIST contains the address of an I/O PCB. The I/O PCB is internally allocated during the schedule process in a DBCTL environment. The I/O PCB is normally used for output messages or to request control type functions to be processed. The PCBLIST and the PCBs reside in a contiguous storage area known as UPSTOR. If the PSB was generated with LANG=PLI, the PCBLIST points to pointers for the PCBs. If LANG= was not PLI, the PCBLIST points to the PCBs directly.

IMS Request

This request makes a IMS or Fast Path database request against the currently scheduled PSB.

You must fill in the following input fields in the PAPL:

Field	**Contents**
PAPLFUNC	PAPLTFUN.
PAPLSFNC	PAPLDLI, DL1 request subfunction code.
PAPLCTOK	DRA request token (output from an INIT request).
PAPLCTK2	Thread Token number 2. This is the DRA request token that is part of the output from a SCHED request.
PAPLTTOK	Thread token set up by the CCTL.
PAPLRTOK	RTOKEN A 16-byte UOR token. See *IMS/ESA Application Programming: Database Manager* for more information about UORs.
PAPLCLST	The address of an IMS call list. See *IMS/ESA Application Programming: Database Manager* for call list formats.
PAPLALAN	Application language type. This must reflect how the call list is set up. If PAPLALAN=PLI, the DRA expects the call list to contain pointers to the PCB's pointers. For any other programming language, the DRA expects direct pointers. PAPLALAN does not have to match PAPLPLAN which schedules request returns. For example, if PAPLPLAN=PLI, the PCBLIST in UPSTOR points to an indirect list. If desired, the CCTL can use this to create a PCBLIST that application programs use. If the application programs are written in COBOL, the CCTL may create a new PCBLIST without pointers as long as the new list actually points to PCBs in UPSTOR. The application program IMS call lists can specify PAPLALAN=COBOL, and the DRA will not expect pointers in the call list.

The following output fields are returned in the PAPL to the CCTL:

Field	**Contents**
PAPLRETC	Code returned.
PAPLSEGL	Length of data returned.

SYNTERM Request

This is a single-phase sync-point request to commit the UOR. It also releases the PSB.

You must fill in the following input fields in the PAPL:

Field	Contents
PAPLFUNC	PAPLTFUN.
PAPLSFNC	PAPLSTRM, sync-point commit/terminate subfunction code.
PAPLCTOK	DRA request token (output from INIT request).
PAPLCTK2	The thread request token number 2. This DRA token is the output from the SCHED request.
PAPLTTOK	The thread token set up by the CCTL.
PAPLRTOK	A 16-byte UOR token (RTOKEN). For information on UORs see *IMS/ESA Application Programming: Database Manager*.

You can also specify the following, optional input fields:

Field	Contents
PAPLSTAT	Address of an area where transaction statistical data is returned to the CCTL.

The following output fields are returned in the PAPL to the CCTL:

Field	Contents
PAPLRETC	Code returned.
PAPLSSCC	State of the single-phase sync-point request at the time of the thread failure. This field is set if PAPLRETC is not equal to zero.
PAPLSTAT	The address of the transaction statistical data area. The address must be specified on the input field.

PREP Request

This is a phase 1 sync-point request that asks DBCTL if it is ready to commit this UOR.

You must fill in the following input fields of the PAPL:

Field	Contents
PAPLFUNC	PAPLTFUN.
PAPLSFNC	PAPLPREP, sync-point prepare subfunction code.
PAPLCTOK	DRA request token (output from an INIT request).
PAPLCTK2	Thread Token number 2. This is the DRA request token which is output from a SCHED request.
PAPLTTOK	The thread token set up by the CCTL.
PAPLRTOK	A 16-byte UOR token (RTOKEN). See *IMS/ESA Application Programming: Database Manager* for more information about UORs.

The following output fields are returned in the PAPL to the CCTL:

Field	Contents
PAPLRETC	Code returned.
PAPLSTCD	Fast Path status code. If the value in the PAPLRETC field is decimal 35, the PAPLSTCD field contains a status code that further describes the error.

COMTERM Request

This is a phase 2 sync-point request to commit the UOR. It also releases the PSB. You must issue a PREP request prior to issuing a COMTERM request.

You must fill in the following input fields in the PAPL:

Field	Contents
PAPLFUNC	PAPLTFUN.
PAPLSFNC	PAPLCTRM, sync-point commit/terminate subfunction code.
PAPLCTOK	DRA request token (output from an INIT request).
PAPLCTK2	Thread Token number 2. This is the DRA request token, which is output from a SCHED request.
PAPLTTOK	The thread token set up by the CCTL.
PAPLRTOK	A 16-byte UOR token (RTOKEN). See *IMS/ESA Application Programming: Database Manager* for more information about UORs.

Specifying the following input field is optional:

Field	Contents
PAPLSTAT	Address of an area where transaction statistical data is returned to the CCTL.

The following output fields are returned in the PAPL to the CCTL:

Field	Contents
PAPLRETC	Code returned..
PAPLSTAT	The address of the transaction statistical data area. This address must be specified on the input field.

ABTTERM Request

This is a phase 2 sync-point request for abort processing. It also releases the PSB. It does not require a preceding PREP request.

You must fill in the following input fields of the PAPL:

Field	Contents
PAPLFUNC	PAPLTFUN.
PAPLSFNC	PAPLATRM, sync-point abort/terminate subfunction code.
PAPLCTOK	DRA request token (output from an INIT request).
PAPLCTK2	Thread Token number 2. This is the DRA request token, which is output from a SCHED request.
PAPLTTOK	The thread token set up by the CCTL.
PAPLRTOK	A 16-byte UOR token (RTOKEN). See *IMS/ESA Application Programming: Database Manager* for more information about UORs.

Specifying the following input field is optional:

Field	Contents
PAPLSTAT	Address of an area where transaction statistical data is returned to the CCTL.

The following output fields are returned in the PAPL to the CCTL:

Field	Contents
PAPLRETC	Code returned.
PAPLSTAT	The address of the transaction statistical data area. This address must be specified on the input field.

TERMTHRD (PAPLSFNC - PAPLTTHD) Request

This request terminates the DRA thread.

You must fill in the following input fields of the PAPL:

Field	Contents
PAPLFUNC	PAPLTFUN.
PAPLSFNC	PAPLTTHD, thread terminate subfunction code.
PAPLCTOK	DRA request token (output from an INIT request).
PAPLCTK2	Thread Token number 2. This is the DRA request token which is output from a SCHED request.
PAPLTTOK	The thread token set up by the CCTL.

Specifying the following input field is optional:

Field	Contents
PAPLSTAT	Address of an area where transaction statistical data is returned to the CCTL.

The following output fields are returned in the PAPL to the CCTL:

Field	Contents
PAPLRETC	Code returned.
PAPLSTAT	The address of the transaction statistical data area. This address must be specified on the input field.

PAPL Mapping Format

The PAPL is the parameter list used by the DRA interface. For the latest version of PAPL mapping format, see the IMS.GENLIBB library; member name is DFSPAPL.

Appendix B. IMS Support for Synchronous DB2 to IMS Data Propagation

DB2 Version 3 Release 1 notifies an IMS subsystem that table data has been captured on the DB2 log.

IMS loads the DB2CDCEX exit routine and passes control to DB2CDCEX for processing and migration of table data to IMS. DB2CDCEX receives control in 31-bit addressing mode (AMODE 31) and passes a register save area that is in 31-bit storage (above the 16-megabyte line). Control must be returned to IMS from the exit in 31-bit addressing mode.

Related Reading: For more information regarding synchronous DB2 to IMS data propagation, see *DataPropagator NonRelational MVS/ESA Administration Guide*.

Bibliography

This bibliography includes all the publications cited in this book, including the publications in the IMS library.

- *ACF/VTAM System Programmer's Guide*, SC38-0258
- *BTAM SP Program Summary*, GC27-0599
- *CICS/ESA 4.1 Application Programming Reference* , SC33-1170
- *CICS/ESA CICS-Supplied Transactions*, SC33-1168
- *CICS/ESA Customization Guide*, SC33-1165
- *CICS/ESA Intercommunication Guide*, SC33-0657
- *CICS/ESA Operations Guide*, SC33-0668
- *DataPropagator NonRelational MVS/ESA Administration Guide*, SH19-5036
- *DataPropagator NonRelational MVS/ESA An Introduction*, GH19-5034
- *DataPropagator NonRelational MVS/ESA Reference*, SH19-5039
- *IBM 4700 Finance Communication System: Controller Programming Library, Volume 3: Communication Program*, GC31-2068
- *IBM 4700 Finance Communication System: System Summary* , GC27-2016
- *IBM 4730 Personal Banking Machine General Information*, GC31-2073
- *IBM 4730 Personal Banking Machine Operations Support Manual*, GC31-2560
- *MQSeries for MVS/ESA System Management Guide*, SC33-0806
- *MVS/ESA Conversion Notebook*, GC28-1436
- *DataPropagator NonRelational MVS/ESA An Introduction*GH19-5034
- *MVS/ESA JES3 Conversion Notebook*, GC28-1438
- *MVS/ESA Linkage Editor and Loader* SC26-4510
- *MVS/ESA Programming: Assembler Services Guide*, GC28-1466
- *MVS/ESA System Codes*, GC28-1486
- *Network Program Products General Information*, GC30-3350
- *Network Program Products Planning*, SC30-3351
- *Network Program Products Storage Estimates* SC30-3403
- *OS/390 V1R1 MVS Planning: APPC Management*, GC28-1807.
- *Service Level Reporter Installation and Customization*, SH19-6437
- *SNA Concepts and Protocols*, GC30-3072
- *Systems Network Architecture Format and Protocol Reference: Architectural Logic*, SC30-3112-02
- *Systems Network Architecture: Sessions between Logical Unit Types*, GC20-1868
- *VTAM Customization*, LY43-0068
- *VTAM Operation*, SC31-6549
- *VTAM V4R2 for MVS/ESA Resource Definition Reference*, SV40-1012
- *VTAM V4R3 for MVS/ESA Programming*, SC31-6550

IMS/ESA Version 6 Library

SC26-8725	ADB	Administration Guide: Database Manager
SC26-8730	AS	Administration Guide: System
SC26-8731	ATM	Administration Guide: Transaction Manager
SC26-8727	APDB	Application Programming: Database Manager
SC26-8728	APDG	Application Programming: Design Guide
SC26-8726	APCICS	Application Programming: EXEC DLI Commands for CICS and IMS
SC26-8729	APTM	Application Programming: Transaction Manager
SC26-8732	CG	Customization Guide
SC26-9517	CQS	Common Queue Server Reference
SC26-8733	DBRC	Database Recovery Control Guide and Reference
LY37-3731	DGR	Diagnosis Guide and Reference
LY37-3732	FAST	Failure Analysis Structure Tables (FAST) for Dump Analysis
GC26-8736	IIV	Installation Volume 1: Installation and Verification
GC26-8737	ISDT	Installation Volume 2: System Definition and Tailoring
SC26-8740	MIG	Master Index and Glossary
GC26-8739	MC	Messages and Codes
SC26-8743	OTMA	Open Transaction Manager Access Guide

SC26-8741	OG	Operations Guide
SC26-8742	OR	Operator’s Reference
GC26-8744	RPG	Release Planning Guide
SC26-8767	SOP	Sample Operating Procedures
SC26-8769	URDB	Utilities Reference: Database Manager
SC26-8770	URS	Utilities Reference: System
SC26-8771	URTM	Utilities Reference: Transaction Manager

	Supplementary Publications	
GC26-8738	LPS	Licensed Program Specifications
SC26-8766	SOC	Summary of Operator Commmands

	Online Softcopy Publications	
LK3T-2326	CDROM	IMS/ESA Version 6 Softcopy Library
SK2T-0730	CDROM	IBM Online Library: Transaction Processing and Data
SK2T-0710	CDROM	MVS Collection
SK2T-6700	CDROM	OS/390 Collection

Index

Readers' Comments — We'd Like to Hear from You

IMS/ESA
Customization Guide
Version 6

Publication No. SC26-8732-05

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? ☐ Yes ☐ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name	Address
Company or Organization	
Phone No.	

Readers' Comments — We'd Like to Hear from You
SC26-8732-05

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department BWE/H3
P.O. Box 49023
San Jose, CA
95161-9945

Fold and Tape | **Please do not staple** | Fold and Tape

SC26-8732-05

Cut or Fold
Along Line

Program Number: 5655-158

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC26-8732-05

Spine information:

IMS/ESA

IMS/ESA V6 Customization Guide

Version 6